IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants:	Airey et al.	Examiner:	Wang, Jin-Cheng
Serial No.:	09/614,363	Art Unit:	2628
Date Filed:	July 12, 2000	Conf. No.:	2211
Title:	DISPLAY SYSTEM HAVING FLOATING POINT RASTERIZATION AND FLOATING POINT FRAMEBUFFERING		

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

DECLARATION OF JOHN M. AIREY UNDER 37 CFR §1.132

Dear Sir:

In support of the accompanying response to the February 20, 2008 office action in the above-referenced application, I hereby declare as follows:

1. I am the first-named joint inventor of the subject US patent application (hereinafter "the Application"), which is assigned serial number 09/614,363 and was filed on July 12, 2000.

2. I am submitting this declaration to show that some details of my invention claimed in the Application are described in US patent number 6,567,083 (hereinafter "the Patent"). In other words, using the language of MPEP 715.01(c), the Patent is a publication of my own invention. To that end, this declaration includes facts showing that:

- I jointly made the invention upon which the relevant disclosure of the Patent is based,
- I was associated with Daniel Baum, the first named inventor of the Patent, and

- Mr. Baum derived his knowledge of the relevant subject matter from me.

I jointly made the invention upon which the relevant disclosure of the Patent is based

3. In 1996, I was employed at Silicon Graphics, Inc. ("SGI") in Mountain View, CA as a staff engineer. In the summer of 1996, I started work on building a machine that used floating point rasterization, including floating point scan conversion, and a floating point framebuffer. The internal code name for the machine at SGI was "*Bali*." The Application describes and claims various aspects of *Bali*.

4. The general architecture of *Bali* is described in the document entitled "Bali Offsite," which is dated November 12, 1996 and attached as Exhibit A. This document was discussed among SGI personnel at an off-site meeting on November 12, 1996. Among other things, this document shows:

- A reference to the "R Chip" on the Agenda page (immediately following the cover page), the Contents page, and page 22. The R Chip refers to the portion of the *Bali* device performing rasterization,
- A chart of software simulating *Bali* on page 21,
- A block diagram of the rasterizer (*i.e.*, the R Chip) and scan converter on page 24,
- a block diagram of a floating-point block on page 29, and
- floating-point multipliers and floating-point normalizers/adders on page 35.

5. A number of other documents show specific details and aspects of my invention. These documents were prepared either by one of the joint inventors of the Application or by

someone with information that originated from one of the joint inventors. Some of those documents are discussed paragraphs 6 to 13.

6. "Bali Floating Point Representation" – A document dated December 3, 1996 discussing a 16-bit floating point format and internal variations used inside the R chip pipeline, and submitted herewith as Exhibit B.

7. "Mapping RenderMan to OpenGL" – A document I authored, dated January 24, 1997, that includes software code to be used with *Bali* hardware to have floating point processing through the R Chip pipeline, and submitted herewith as Exhibit C

8. Document containing Code headed "head 1.1" - A document I authored, dated January 24, 1997, that describes rasterization of pixels, color processing using floating point numbers and framebuffers, and submitted herewith as Exhibit D.

9. "Extended Range and/or Precision" - A document dated February 15, 1997 and authored by my co-inventor, Mark Peercy, and me. This document describes calculations using floating point numbers and which format of floating point numbering to use, and submitted herewith as Exhibit E.

10. "Tiny Floating Point" - A document dated February 15, 1997 and authored by Mr. Peercy and me. This document describes disadvantages of fixed point arithmetic and further describes a tiny IEEE like floating point format, and submitted herewith as Exhibit F.

11. Document containing Code headed "head 1.5" - A document with versions ranging from dates from July 15, 1997 to July 28, 1997 and authored by John Paul Alex, who was an intern working for Mr. Peercy and me. This document describes calculations using

floating point numbers and which format of floating point number to use, and submitted herewith as Exhibit G.

12. Email from me to Mr. Baum dated August 14, 1997. In this e-mail, I mention the s10e5 pipeline and high-speed data transfer from the host to the frame buffer, the frame buffer to the texture memory, and texture memory to the frame buffer. This document is attached as Exhibit H.

13. "Invention Disclosure" - A document dated September 22, 1997 that is an invention disclosure submitted to SGI Legal Services, and submitted herewith as Exhibit I. The document describes a 16 bit floating point format for texture store and framebuffer. The document further states that the invention was conceived about a year prior to the submission of the Invention Disclosure. This document also notes that Mr. Baum is the department manager for visual systems. Mr. Baum is not listed as an inventor.

14. Mark Leather, who was a member of the SGI technical staff from 1989-1997 but not an inventor on the Application, further corroborates my invention of the subject matter.

15. During a deposition conducted on December 7, 2007 (see Exhibit J), when asked if it was his "understanding that [the idea of multipassing data through the frame buffer] involved the use of floating point formatted data," Mr. Leather responded: "Yes, that was my understanding."

16. Further, in response to the question "And did you understand that that was [Mark Peercy's] concept that he was working on at SGI?," Mr. Leather responded: "I believe it was him and John Airey."

17. During the deposition, in response to the question "But do you recall the use of a floating point frame buffer as it related to Bali?" Mr. Leather responded: "Yes." Further, in response to the question "And that was Dr. Peercy and Dr. Airey's work?" Mr. Leather responded: "Yeah."

18. During the deposition, Mr. Leather recollected that a second project at SGI had a floating point frame buffer, but when asked if the "other work was an extension of Airey and Peercy's work," Mr. Leather replied: "Yeah."

I was associated with Daniel Baum at the time of my joint invention

19. Mr. Baum was my supervisor as the Hardware Director for the *Bali* project. We thus both worked together at SGI at the same time—among other times, while I was developing the invention claimed in the Application.

Daniel Baum derived his knowledge of the relevant subject matter from me

20. In his capacity as hardware director of the *Bali* project, Mr. Baum was responsible for understanding the technology and, as such, he was a decision-maker on the direction, schedule, and features of the project. In addition, in this capacity, Mr. Baum was responsible for updating company executives. To fulfill these and other responsibilities, Mr. Baum needed to be familiar with the software simulation, the floating-point scan converter, the floating-point frame buffer, and their technical advantages/benefits.

21. My co-inventors and I therefore described various aspects of the invention and the simulation to Mr. Baum so he could make the appropriate decisions. For example, Mr. Baum

was present at the above noted *Bali* offsite meeting of November 12, 1996 (see Exhibit A), as well as other internal *Bali* meetings. In addition, in his capacity as Hardware Director of *Bali*, Mr. Baum had access to many of the documents discussed herein.

22. As noted above, on August 14, 1997, Mr. Baum sent me an e-mail asking about opportunities to differentiate SGI. I responded, as noted above, by discussing the s10e5 pipeline and floating-point frame buffer (see Exhibit H). I thus conveyed details of my invention to Mr. Baum in this e-mail.

23. Danny Loh, one of the joint inventors of the Application, worked on the software simulation that appears to be mentioned in the Patent. Specifically, during a deposition conducted on November 9, 2007, Mr. Loh stated "So the question you asked me before, did I work with John Airey on the floating point on Bali?.... Yes." Mr. Loh further stated that he "wrote sort of the simulation framework to evaluate floating point formats in the frame buffer." When asked if Mr. Loh was doing that work with John Airey, Mr. Loh replied "with John Airey and Mark Peercy." That portion of Mr. Loh's transcript is in the prior noted Exhibit J.

24. In his capacity as Hardware Director, Mr. Baum should have known about the software simulation.

25. It is my understanding that before the filing date of the Patent, SGI did not have another floating-point project that was independently derived. Instead, as Mr. Leather stated under oath, if any such projects existed, they were derived from the principles of my joint invention.

26. I hereby declare that all statements made herein of my own knowledge are true, that all statements made herein on information and belief are believed to be true, and further that these statements were made with the knowledge that willful false statements and the like are punishable by fine or imprisonment, or both under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

John M. Arey

Dated: May 1st 2008

02839/00115 860759.1

poult

EXHIBIT A

Bali Offsite

11/12/96

Arch Cook
Hennessy

Silicon Graphics Inc. Company Confidential

Agenda

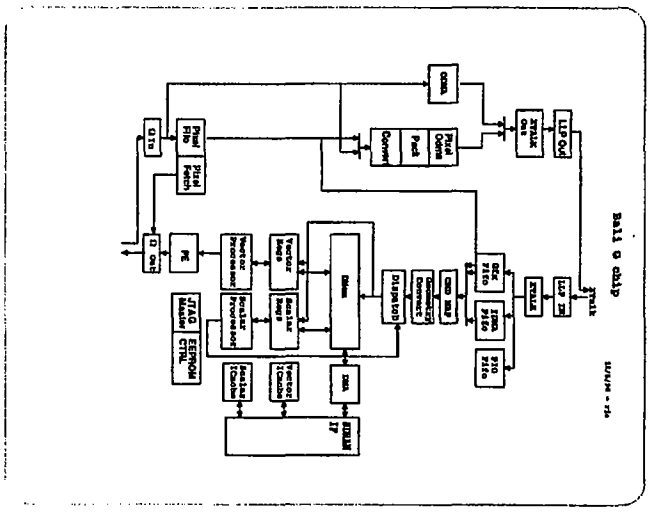
9:00	BREAKFAST
9:15	Introduction
9:16	G Chip
10:45	BREAK
10:55	N Chip
11:15	Documentation Environment
11:45	Design Flow
12:00	LUNCH
12:45	R Chip
2:15	Software
2:30	M Chip
2:50	BREAK - Driving Range
3:30	D Chip
4:20	System
4:50	Wrap-up
5:15	AJOURN

Contents

G Chip	1
GE Ucode	3
GE vector and scalar	6
PE	10
GRU	13
N Chip	14
Documentation Environment	17
Simulation Environment	21
R Chip	22
Scan Conversion	24
Texture	27
Texture Filter	29
Lighting	30
FRU	38
SRAM	44
Pixel Response	48
Pixel SW	52
Geometry Pipeline	56
D Chip	58
System Configs	62
M Chip	65

G

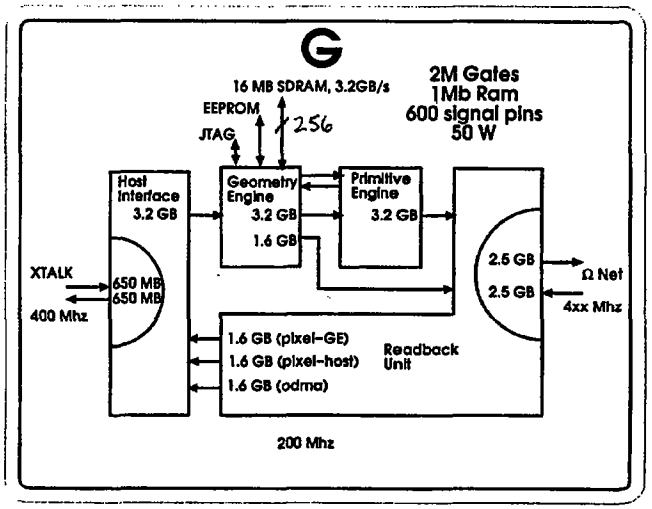
Bob Alfieri
Roger Allen
Paul Askounis
Avi Bielweiss
Gordon Elder
Rob El-Kareh
Efi Fogel
Celeste Fowler
Ziv Gigus
Robert Keller
Gloria Lau
Erik Lindholm
Randy Martin
Atul Narkhede
Reuel Nash
Paul Simonsic
Mark Stadler
Bob Stoll
Paul Thilking
Mark Young



G

G Targets

- 15M (v3f,t2f,n3f)/sec (XTALK limited)
- 30M (v3f,t2s,n3s)/sec (vertex array)
- 40M v3f/sec
- Pixels/Texels at Xtalk rate (650 MB) reads & writes



3 clock domains

2 intf
1 core

G

Big changes:

Simpler vector unit
Commodity Scalar unit

Special needs:

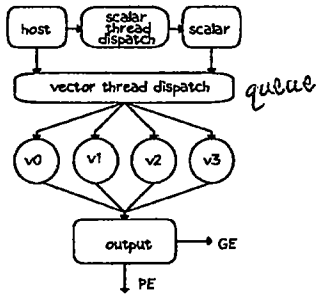
µp core
1kx128 7port 200 Mhz ram
XTALK Net interface

Interfaces:

XTALK, Ωnet, SDRAM, EEPROM, JTAG, NIC

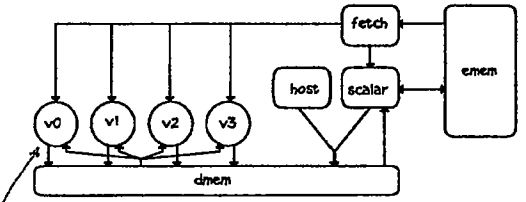
Vector Unit Programming Model

- MIMD of four vector sub units (Kona was SIMD)
- Host commands or scalar unit thread driven
- Round Robin thread distribution (amongst $v_0..v_3$)



Vector Unit Programming Model (cnt'd)

- Unified memory:
  - Internal memory – dmem (16KB):
    - shared across vector and scalar units with a port per
    - quad word data path per vector port
    - accessible by either host or scalar unit
    - resources: command fifo, immediate state and scratch
  - external memory – emem (8MB):
    - instruction memory, four parallel fetch
    - state backup



2 quad word wide

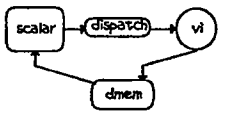
Vector Unit Programming Model (cnt'd)

- Vector thread forks a single or a chain of vector code fragments – inline code or unrolled loops.
- Vector thread context: consists of:
  - vector dmem work pointer (cmd fifo or scratch)
  - instruction entry point, instruction count, indirection
  - special dmem access

  ↳ for stitching several vector code fragments
- Vector generates output to either the PE or the scalar unit (dmem)

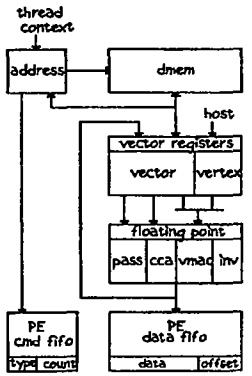
Vector Unit Programming Model (cnt'd)

- Synchronization – since 2 srcs of ctrl (scalar unit, host) uses semaphore in dmem.
  - logical programmable threads
  - scalar generated output reuses dmem cmd fifo area
  - dependencies:
    - host and scalar to dispatch
    - vector to output
    - vector to scalar



Transpose a 4x4 region of mem

Vector Unit Programming Model (cnt'd)



- Load/store RISC model (transposed load/store for surfaces and matrix operations)
- Direct and indirect (array of) pointers
- Persistent vertex state
- Vector lengths of 1 to 4 elements
- Single precision vector floating point operations: scof, dot , cross
- Generic conditional register move

scof - scale & offset

Vector Unit Programming Model (cnt'd)

- Instruction set: 128b instr word
  - four parallel control fields: load, store, floating point and sync
  - load immediate of memory address, store base address + offset
  - operand negation
  - arithmetic ops – all under vector mask:
    - three operands: scof (A*B+C),
    - two operands: dot (A*B -> scalar), cross
    - single operand: pass, inv, invsq
  - condition code:
    - per element conditional move
    - persistent condition code
    - condition code accumulation – frustum clip chk.
  - scalar to vector spread

cca - cond code for clip. a*b+c

Vector Unit Programming Model (cnt'd)

- Code examples:
  - load an immediate address, scale offset operation, store a quad word to memory:

```
lda a0, GL_MV_MAT; \
scof a(v0), b(v1), c(zro), r(v10); \
stq b0(v5), 0(a1);
```

  - load quad word from memory, two element dot product, scalar to vector spread:

```
ldq b0(v12), 0(a12); \
dot vm(0x5), a(v3), b(v17), r(v10, d1);
```

  - cross product, conditional move of vector elements (f – false):

```
cross vm(0x6), a(v3), b(v14), r(v10);\
@ cc 1(f), 2(f), 3(f);
```

Geometric Statistics

- Ran across seven representative applications/demos
- Most typical mesh size used: 4.
- Xform state changes are at about once per mesh in a non flattened CAD application
- Material calls reached couple of hundreds in a frame
- No use of texgen

wasted per vertex lighting vs per pix.

Geometry Performance

- Host to perform tasks beyond the scope of transport layer
- Immediate (PIO) vs list (DMA)
- Use mesh size of 4 for performance evaluation
- Vertex array as primary transport model
- Raise up bar for canonical benchmarks: default to normal normalization and texture xform

5

Geometry Performance (cnt'd)

- Initial figures based on vector ucode cycle count (cpu - 550Mhz Beast, DMA - Duplo Net 1.2GB/sec)

  Notation:
  - vertex0: no light no texture (v3f)
  - vertex1: infinite light and texture (t2f,n3s, v3f)
  - vertex2: local light, local viewer, no texture (n3s,v3f)
  - vertex3: local light, local viewer and texture (t2f, n3s, v3f)

  - eval mesh: bi cubic, 2 vertices per vector unit, autonormal, vertex processing
  - eval coord: bi-cubic 1 vertex per 2 vector units, autonormal, vertex processing

Type	CPU (lm) (MV/sec)	DMA(list) (MV/sec)	1V (cycles)	4V's (MV/sec)	CPV (cycles)
vertex0	48.8	85.7	21	38.0	5
vertex1	27.1	37.5	28	28.5	7
vertex2	34.3	50.0	37	21.5	9
vertex3	27.1	37.5	40	20.0	10
eval mesh	na	na	200	8.0	25
eval coord	na	na	400	2.0	100

Geometry Performance (cnt'd)

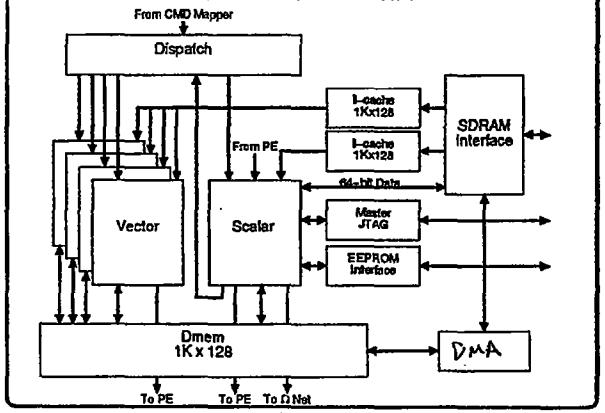
- One thread per vector sub unit
- Performance estimates meet our goal but...
- Risk - aggressive assumption on floatin point latency
- Initial stats:
  - vector to CPU ratio is between .77 - 1.05
  - vector to DMA ratio: .44 - .76

4x & 6x speedup for vertex + evaluators respectively over 4 kona.

Scalar Unit Requirements

- 200 Mhz operation
- Floating point support for per-vertex lighting and clipping support. + lot stuff for compression.
- C compiler, assembler, and debugger available. C-based simulator desirable.
- Verilog models available (behavioral and timing-accurate) for simulation.
- High bandwith data path (64-bit) for Dmem interface.

GE – Vector+ Scalar Units



Scalar Unit Options

Option 1:

Texas Instruments has a 200 Mhz DSP core that looks like it would meet our requirements.

Pro:

- The C7x performance numbers are:
  1600 MIPS @ 200Mhz
  800 MFLOPS single-precision @ 200Mhz
  300 MFLOPS double-precision @ 200Mhz
- The C7x would be an on-chip solution with fast access to Dmem.

Con:

- TI plans on late Q2 '97 tape-out. Some risk if they slip.
- Unlikely option if TI is not our ASIC vendor.

Scalar Unit Options

Option 1a:

A processor core provided by our ASIC vendor.

Pro:

- Would provide a design flow integrated with the vendor.
- Would be an on-chip solution.

Con:

- Current likely vendors (other than TI) do not seem to have a solution that meets the requirements.

Ball Offsite II

Scalar Unit Options

Option 2:

A core provided by a third party design house, internal MIPS core, or home-grown design. Includes using TI with a different ASIC vendor

Pro:

- Would be an on-chip solution.

Con:

- Would have to find reliable third party design house that we could work out a design flow with.
- An internal MIPS port or home-grown design has increased complexity and risk.
- TI has not decided yet about licensing their core to another vendor

Ball Offsite II

Scalar Unit Options

Option 3:

Off the shelf micro-processor external to the G chip.

Pro:

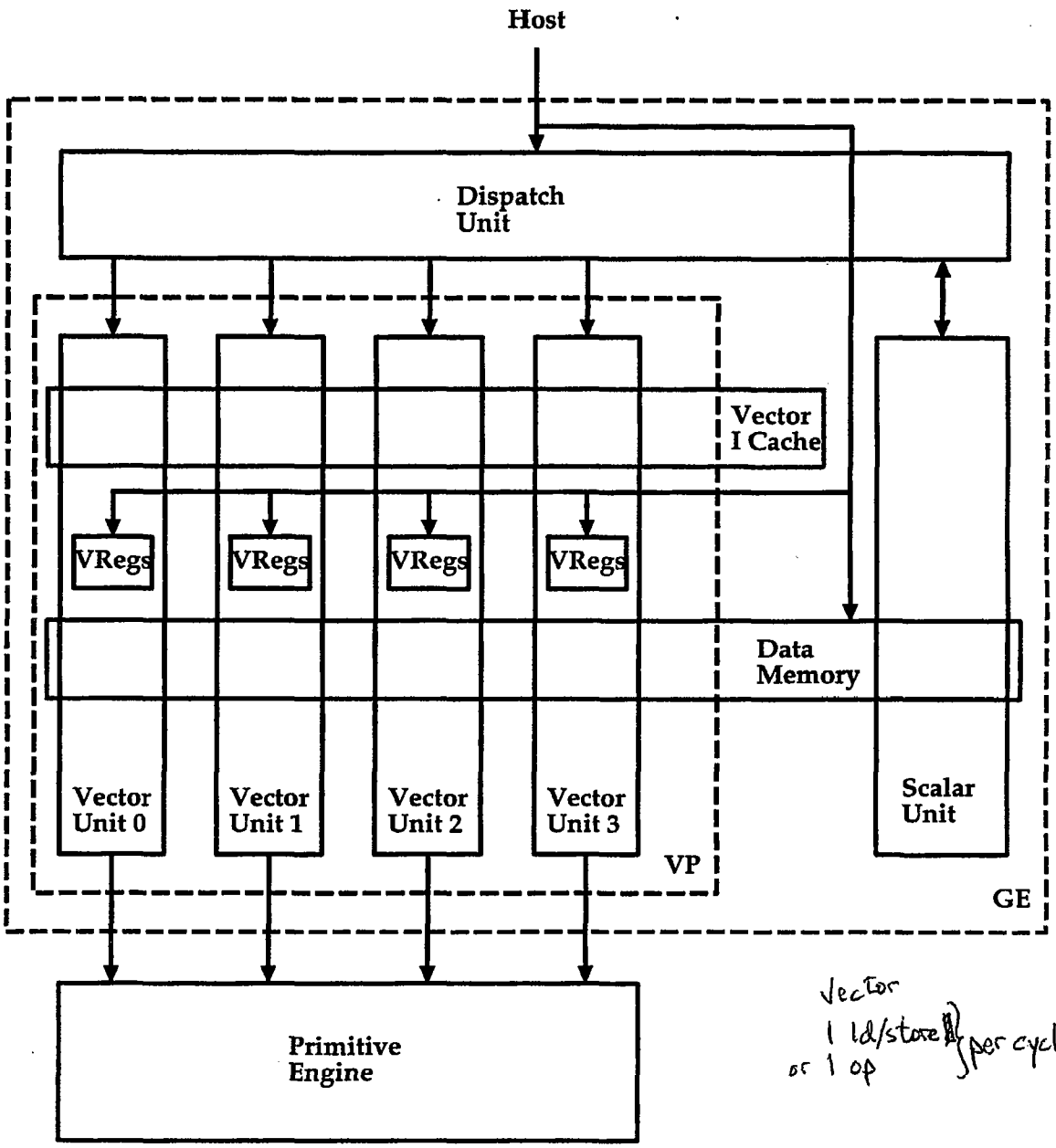
- Processors with 200Mhz operation and floating point are available.
- Off the shelf micro-processor probably has better (bigger) cacheing system than on-chip solution.
- Low risk for obtaining chips and development tools.

Con:

- Data rate to/from Dmem affected due to connection thru a 64-bit muxed address/data bus running at 100Mhz
- Increased pinout over internal solution. Increased cost?

7

Vector Processor



MIMD

Vector
1 ld/store } per cycle
or 1 op

VP	= 1M gates,	170K bits,	272 sq mm
VU	= 250K gates,	9K bits,	34 sq mm
DMem	= 5.8K gates,	384K bits,	24.4 sq mm
VI$	= 6.3K gates,	134K bits,	15.34 sq mm

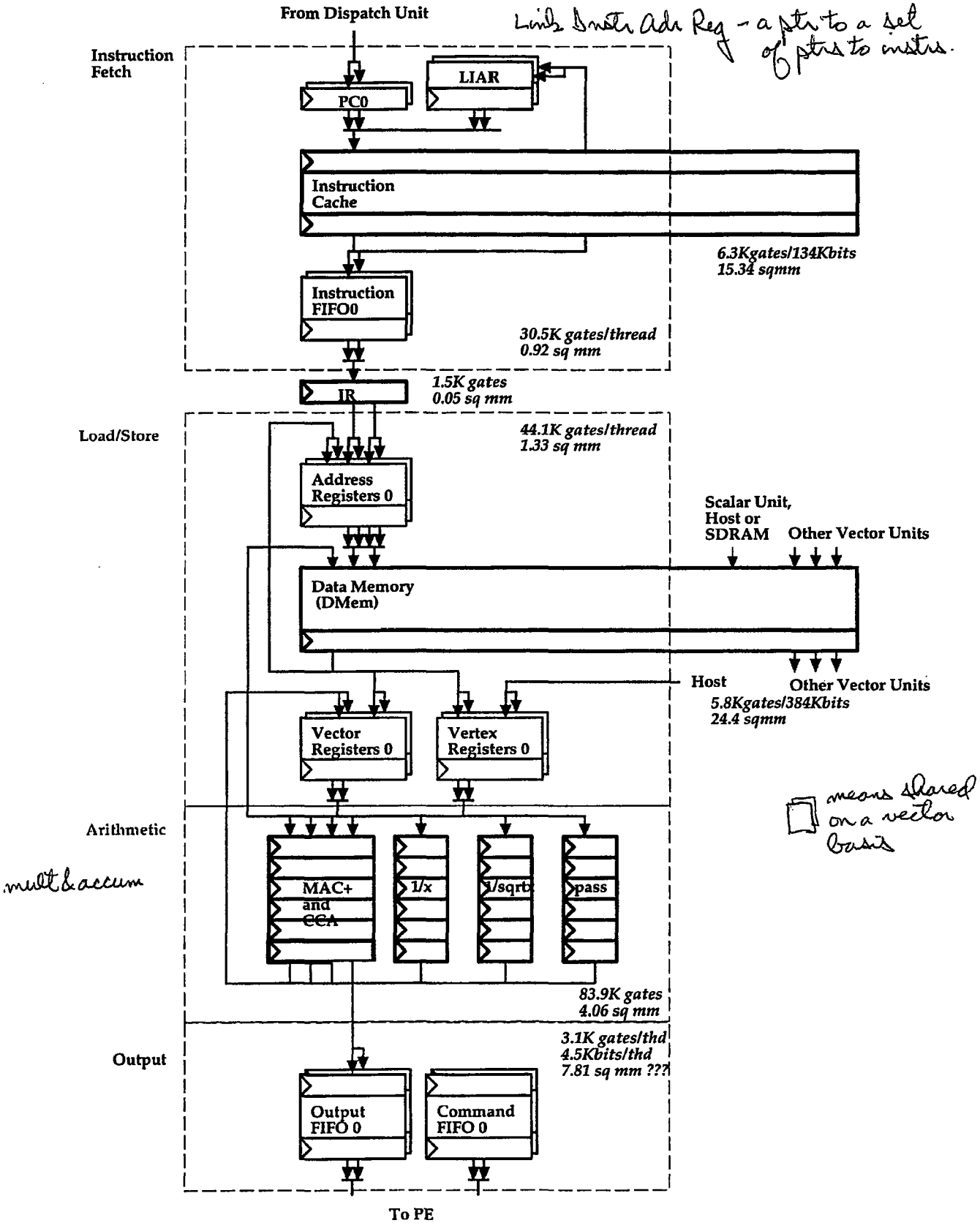
SGI Confidential

8

rog 11/9/96

Vector Unit Pipeline

250K gates for 2 threads
34.4 sq mm



SGI Confidential

rog 11/9/96

PE Design Notes

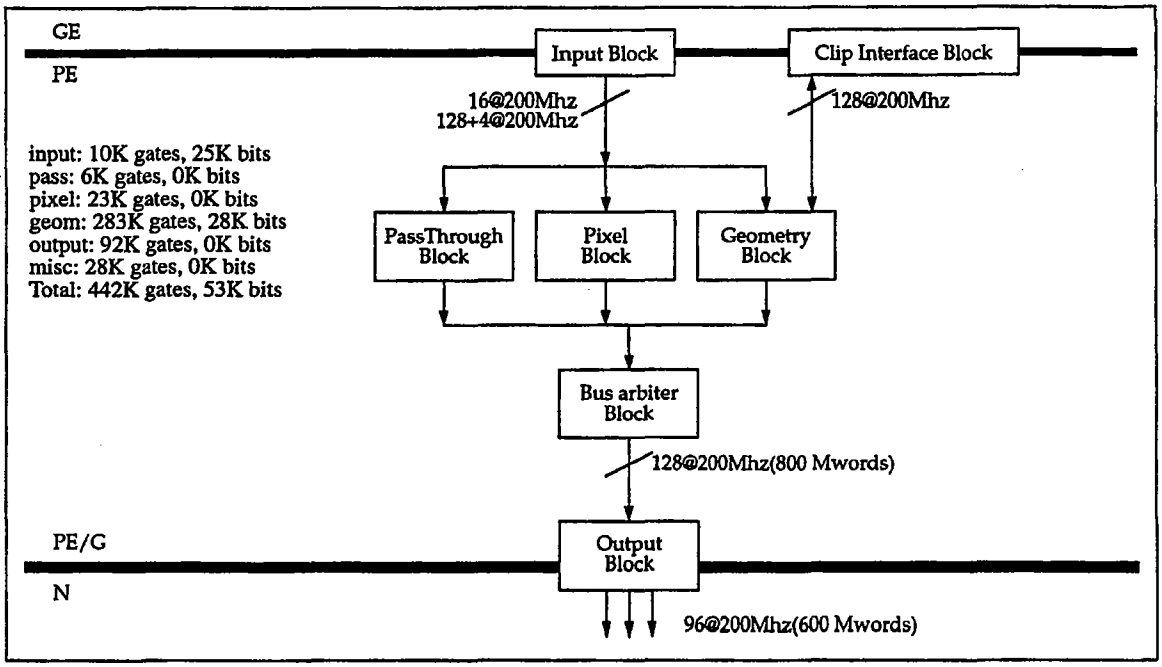
Rob (Skip) El-Kareh

Gloria (Globot) Lau

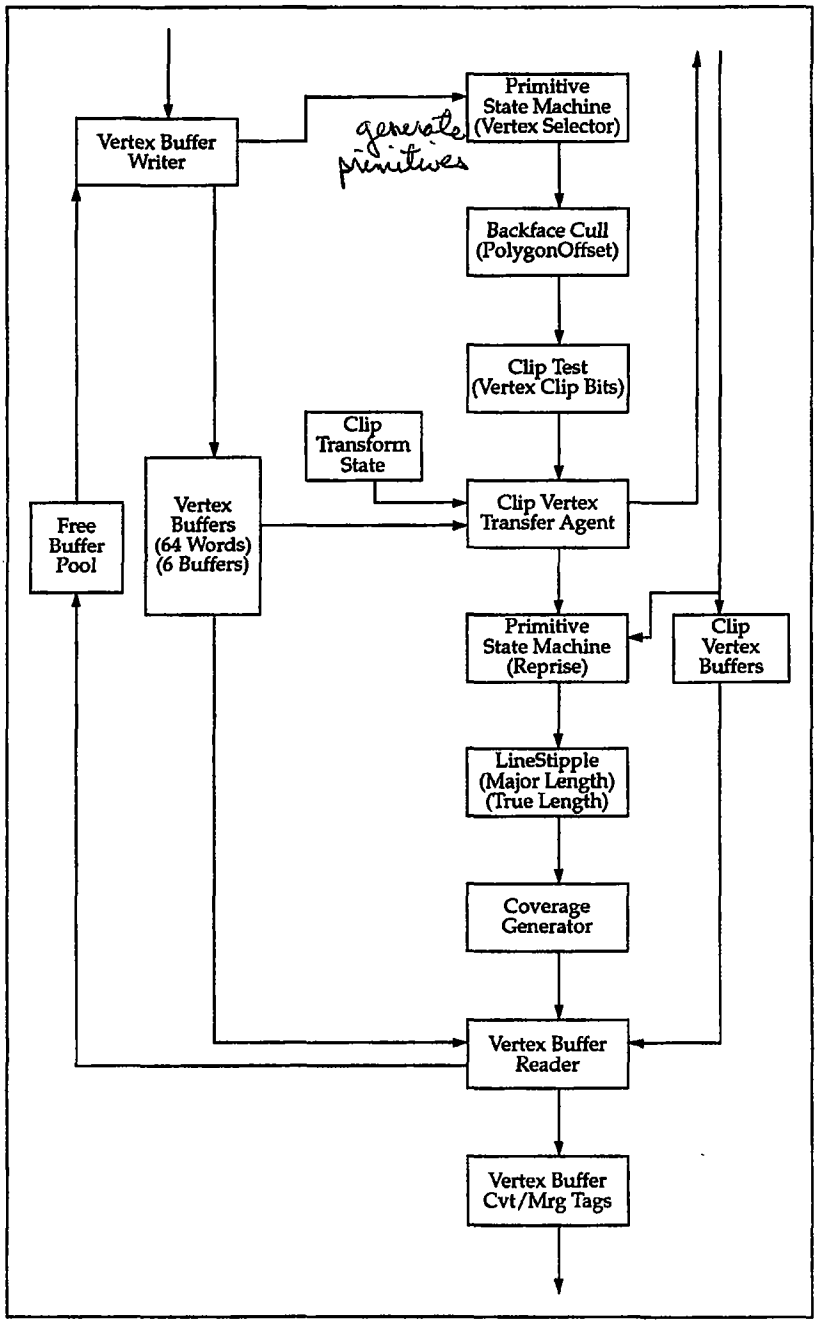
Erik (Bowzer) Lindholm

Paul (Ironman) Thilking

Mark (Flyboy) Young



PE / Top Level Block Diagram



Geometry Block / Internal Block Diagram

PE Functional Goals

- GE interface to Omega Network
  - Includes interface to Convert/Merge
- Pixel conversion, reformatting, and distribution
  - Input (IEEE Float, U16, U8), start aligned, no garbage
  - Output (S10E5, U16, U8), first pixel word aligned start of each packet
    may happen in two in GE
- Bitmap unpacking, reformatting, and distribution, non-opaque and opaque, pixel packet per bit

- Geometry primitive processing and distribution
  - Standard OpenGL primitives (with swap)
    tmesh for Iris GL
  - PolygonMode, line and point generation
  - PolygonOffset, dz/dx & dz/dy, with scale and bias (may incur slight penalty)
  - LineStipple counter, major and true lengths, with first vertex of each segment
  - Clip exception processing (sent to GE, WILL incur significant penalty)
    ? Want extra info w/vertices. So we can. Ship it back in this case.
  - Backface cull before clip
  - Zmin/Zmax/Hitflag for select
  - Bounding box coverage generation per primitive (including lines)
  - Feedback on Host or in GE } Maybe handled in GE or Host

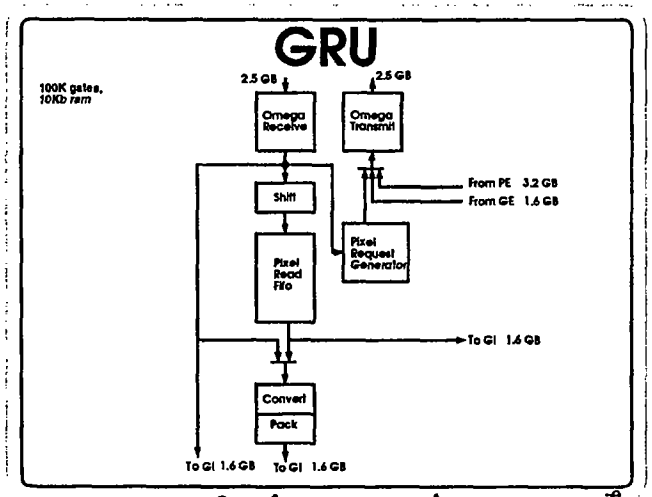
- Rasterizer State Management
  - Write / WriteThrough
  - Dirty Bits per State Set
- R Chip Synchronization
  - Broadcasts SyncID to R chips when Geometry is received before SyncFlag is cleared

Used for texture download

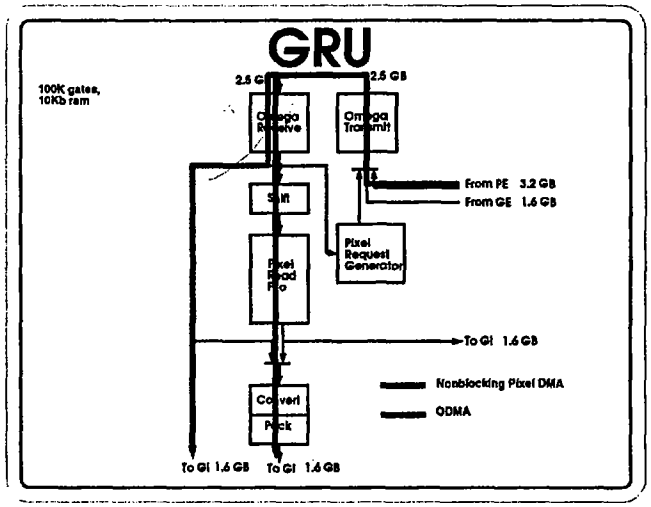
- Assumptions
  - R Handles: stippled line rasterization, wide stippled lines, wide points, AA lines and points
  - Bounding Box rasterizer selection is good enough for lines
- Open Issues
  - Who handles rasterization of Constant Multisample Points (Lightpoints)
  - What is the true benefit of R state shadowing? What is the state size? Can it be grouped into sets? Could it be better managed in N Chip?

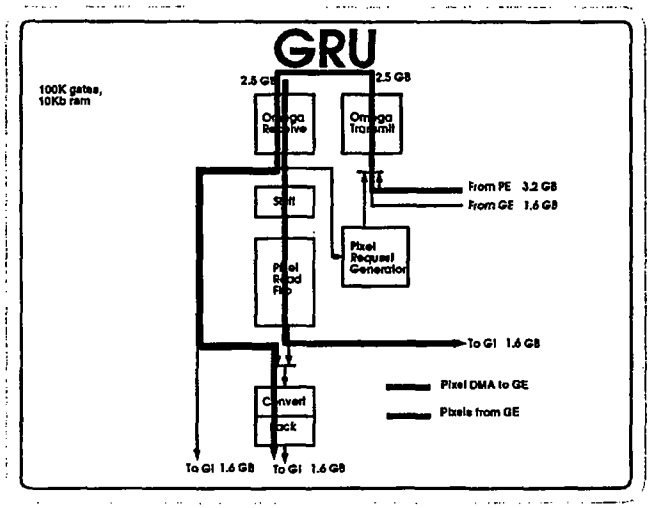
Material changes for per vertex lighting ...?

pixel readback

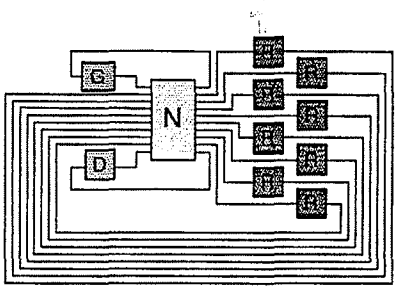


GE → host goes out omega net & back (saves pins)



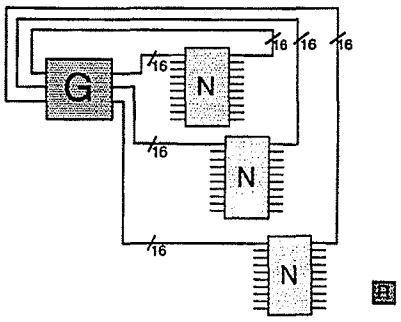


10-Node Configuration

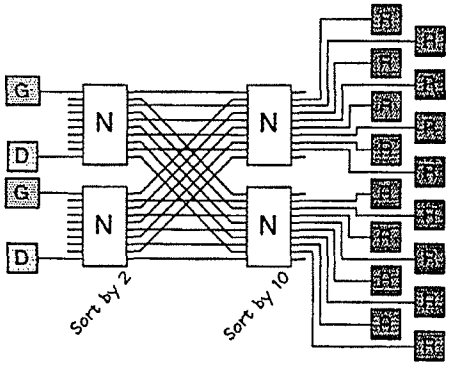


Was single 486 net.

Three 16-bit Channels into and out of each Node



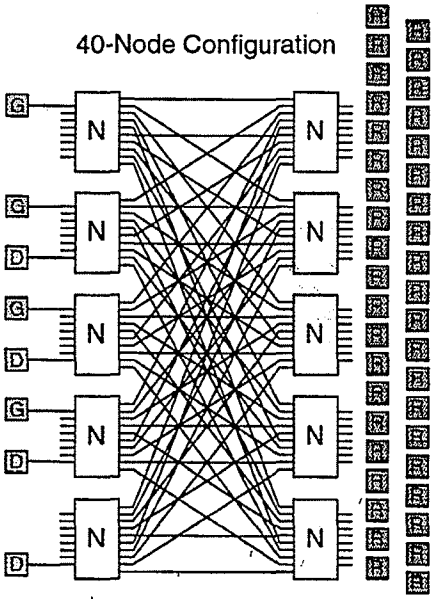
20-Node Configuration



12 N chips.

40-Node Configuration

32 R's



notice imbalance of ios.

Could use 4 chips, but no fair division.

SGI Confidential

N-Chip Issues

* **Bandwidth and Latency**
  We can get more bandwidth with deeper FIFOs.
  We can get better latency with shallower FIFOs.

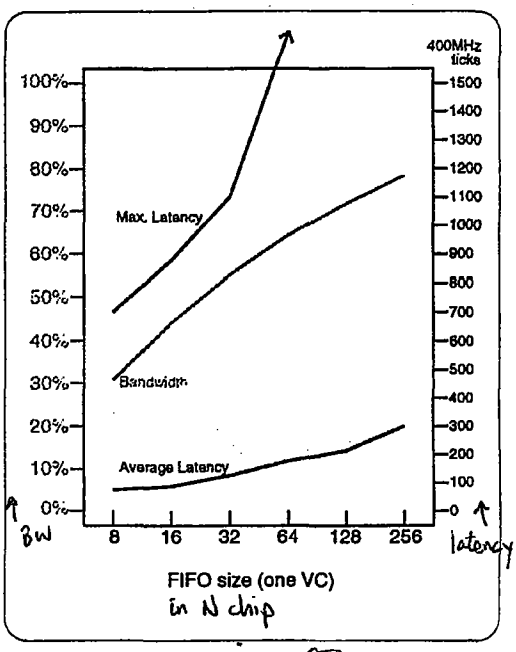
* **Packet Ordering**
  N-Chips can guarantee order within a channel, but there is no coordination between channels.

* **System-level Deadlock**
  Example: network clogged by texture requests so it cannot send texture responses.

  One solution: virtual channels...one class can go when the others are blocked.

* **Flow Control**
  Throttle wire or credit scheme.
  Will need separate one for each virtual channel.

* **Error Handling**
  At least want error detection.
  Correction: ECC or retry protocol.



Large # of tiny FIFOs.
Small # "large"

Packet Ordering

Only one path from Node A to Node B within Omega network, so packets on any one channel arrive in order.

But there is no ordering control (and wide latency variation) between 16-bit channels.

Most transfers are self-descriptive and do not care about ordering:
- Texture requests
- Texture responses
- Video requests
- Video responses
- R → G Flow control

Others will have to cope using sequence numbers.

Because of latency variations, better to resequence after buffering...no waiting!

System-Level Deadlock

As network clogs, we need a way to relieve backpressure.

Must give precedence to "downstream" messages (e.g., responses instead of requests).

Virtual channels (sharing the same wires) is traditional way to allow high-precedence messages to flow despite snarl of low-precedence messages.

Virtual channels imply:
- Separate buffering
- Separate flow control
- Intermingling between virtual channels
  (each transfer identifies its virtual channel)

We need to identify all possible deadlocks!

We need to categorize packets into virtual channels to assign precedence.

Example:

High	Medium	Low
Video request	Texture request	Triangles
Video response	R → G Throttle	Vertexes
Texture response	Pixel read data	Read requests

Flow Control

Sometimes node cannot accept packets.
Example: R-chip can be overwhelmed by texture requests or pixel reads or triangles.

Need to be able to refuse packets.

Triangles handled by higher-level mechanism:
Throttle packets from R→G.

Others handled by flow control wires. + one per VC.

Two schemes:
Throttle: shut up ASAP.
Credit: keep track of receive buffers

Credit is more effective...hides the latency...but it presumes one set of receive buffers. If we have multiple FIFOs behind each input, credit scheme gets complicated.

Throttle scheme always works, but must set high-water-mark lower to skid due to latency.

Error Handling

Source-synchronous signalling can have errors unless it is tuned perfectly.

During bringup, "perfect tuning" can take months to achieve. Error correction can allow this effort to proceed in parallel with debug, not in series.

At least two possible schemes:
Error-correcting codes
Retry

Before we choose, we need to understand what kind of errors we expect.

ECC requires less hardware, but fixes fewer errors. It is well-suited to single bit errors, not bursts.

Retry requires a lot of hardware...big send buffers. More robust in case of many errors.

In both cases, handling errors at a finer grain requires less RAM...correct or retry micropackets..

N Chip Estimate

10 input ports, 10 output ports, 16–bit/port

Die size:	14mm x 14mm
Total standard cell gates:	325.6K gates
Total ram bits:	125.0K bits
Total signal I/O:	~480 pins
Total package I/O:	~900 pins
Core frequency:	143MHz

Input unit (SSR, cmd decode, flow control)
Error handling (error detection and correction)
Performance monitor
diagnostic logic
Output unit (SSD, output arbitration)
Virtual channel0 fifo (100 x 40 x 16)
Virtual channel1 fifo (100 x 40 x 16)

Ball Documentation Methodology

Mark Leather

Silicon Graphics – Confidential

How was Kona documented?

- Initial architecture documentation was done in showcase
- This was later turned into (official) framemaker spec documents for each ASIC
- Consistancy of style was achieved through a published set of guidlines
- Templates/style sheets were made available to help enforce guidlines
- Externally generated documentation was imported by hand into framemaker
- Web site created using third party frame–to–html utility

Silicon Graphics – Confidential

Kona register spec language

- *Mark Grossman proposed a database language for describing registers*
- Utilites were written to parse this language and create include files for C and verilog
- Mark Leather later extended this to generate a web site containing drawings and descriptions of each register and field
- The register web site was very heavily used
- *Incorporating this into framemaker documents created a problem.* Register drawings needed to be converted to eps and imported into framemaker. Register and field descriptions needed to be re–written by hand.

Silicon Graphics – Confidential

What did we learn?

- *Formatting guidlines were only loosly adhered to*
- Framemaker documentation took up a considerable amount of the designers time
- Much of the documentation was done after chip tapeout, when it was least needed
- Showcase documentation was poorly organized and difficult to find
- Web conversion mechanism sometimes did not work
- Importing the register stuff took so much work that it somewhat nullified the original objective.
- Everyone hates framemaker!
- Whole documentation process consumed vast amounts of paper

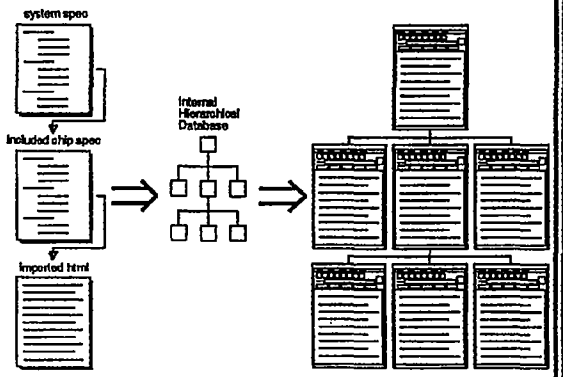
Silicon Graphics – Confidential

Ball documentation methodology

- Extend register definition language to cover everything
- Ball documentation will consist of a hierarchical database containing node types for registers, interfaces, chips, chip blocks, algorithms etc.
- Database spec format will be ASCII
- Multiple levels of include will be supported such that each designer can work on his/her part of the database.
- A node linking facility will prevent duplication of work wherever possible. e.g. the description of "Z Buffering" only needs to be entered once.
- Any fields requiring more than 2–3 lines of text or HTML, can be imported, allowing the use of existing HTML editors such as Webmagic, Cosmo Create or Navigator Gold.
- Database used as input to a new tool which will create a complete web site

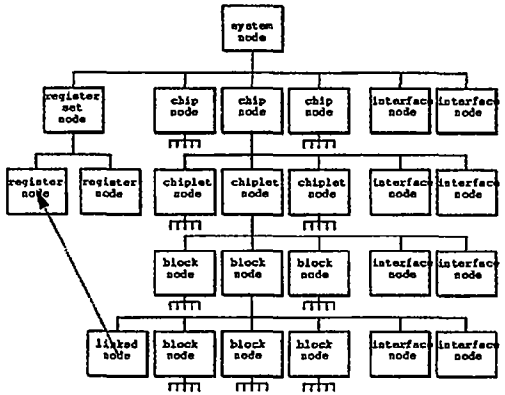
Silicon Graphics – Confidential

Ball documentation process



Silicon Graphics – Confidential

What does a typical documentation database look like?



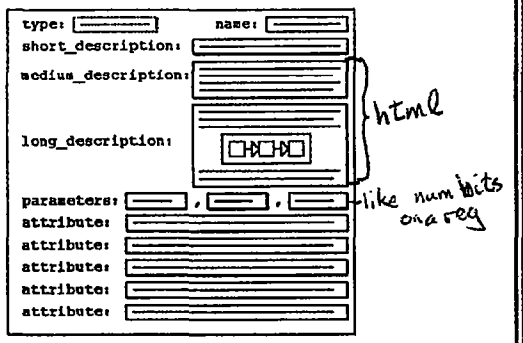
Silicon Graphics – Confidential

Node types currently supported

`system:`	This describes the complete system
`board:`	This describes a board within the system
`chip:`	This describes an ASIC within the system
`chiplet:`	*This describes an ASIC partition entirely owned by one design team*
`block:`	This describes a smaller ASIC partition, usually related to a specific verilog module in the design hierarchy
`interface:`	This describes a set of wires linking two or more blocks, chiplets or chips
`feature:`	This describes, in algorithmic terms, a specific feature of a block, chiplet, chip or system (such as Z buffering).
`packet_grp:`	This describes a set of multi-clock packets belonging to a given interface
`packet:`	This describes a specific packet in the packet group
`register_set:`	This describes a set of registers belonging to the same address space
`register:`	This describes a specific register in the register set
`field:`	This describes a field of an interface, packet or register
`misc:`	This is a general node reserved for information which does not fit in any of the above categories. New node types may get added in order to better categorize this information as required.

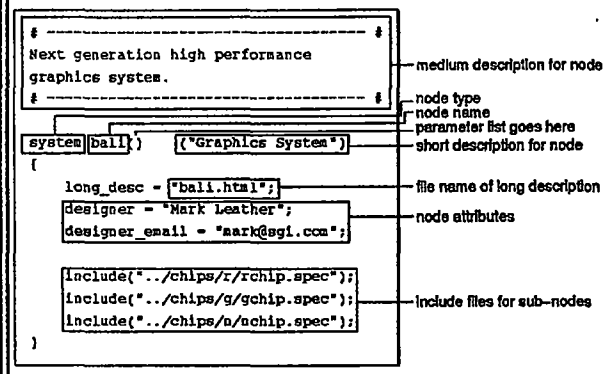
Silicon Graphics – Confidential

What does a database node look like?



Silicon Graphics – Confidential

What does a typical spec file look like?



Silicon Graphics – Confidential

How do we allow the spec language to evolve

- The language should be general enough to allow new node types and node attributes to be added or removed at any time during the project cycle
- It is very important not to break the document tree when new features are added
- Node attributes are assigned a priority – "optional", "recommended", "required" or "not recommended". A missing attribute generates a warning if "recommended", an error if "required". A specified attribute generates a warning if "not recommended".
- New required attributes are initially prioritized as "recommended"
- Attributes to be deleted, are re-prioritized as "not recommended"
- Wait one or more weeks until the documentation is warning free – then upgrade the new attribute to "required", or remove old attribute
- A web site will give a dated list of all spec language changes

Silicon Graphics – Confidential

Printed output

- Printed output should be made possible, but actively discouraged
- Requires a book spec file – this sequentially lists all nodes to be printed
- Printed output will be in the form of a book, consisting of a list of contents, and a set of pages, containing a list of cross references at the bottom of each page.
- All hyperlinks will be converted into cross references. A cross reference consists of a page number if the reference exists in the current book, a list of book titles and page numbers, if the reference exists in other books, and a URL for the online version of that reference.

Silicon Graphics – Confidential

Getting Started

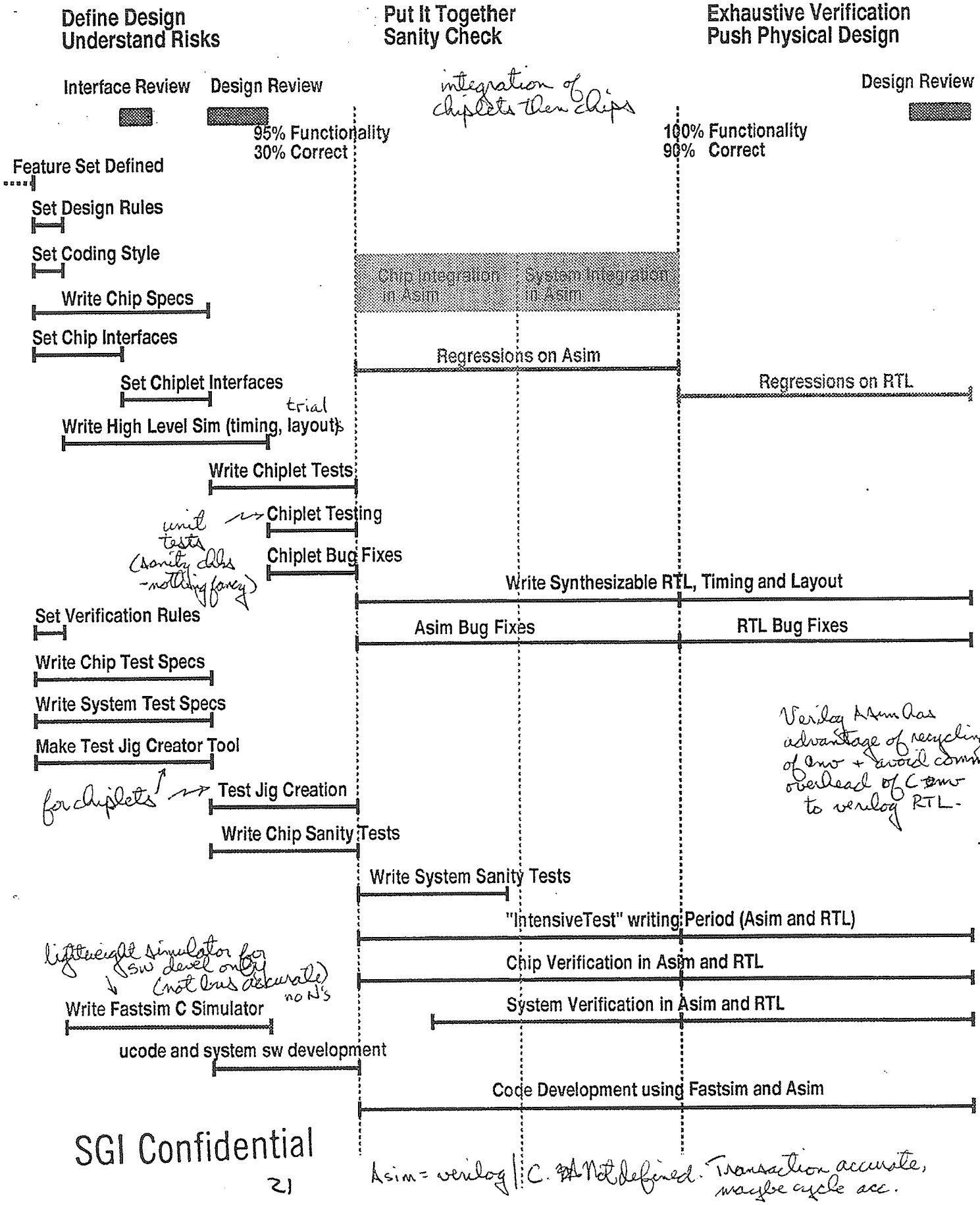
- Official Bali Documentation Web Site
  Look for this under the Bali Web Site in the next few days
- Language specification
  Look for a fully hyperlinked language spec in the Bali documentation site
  Until this is available, a text based spec is available on the Bali tree under
  `gfx/BALI/tools/nreg/spec`
- Example spec file
  An example spec file is available on the Bali tree under
  `gfx/BALI/tools/nreg/test`
  The resulting generated web site can be seen in
  `http://tequila/rtmp`
- HTML generation software
  This can be found on the Bali tree under
  `gfx/BALI/tools/nreg/`
  The usage is
  `nreg <specfile> -h <destination_dir> <destination_url>`

Silicon Graphics – Confidential

20

Trial layouts, etc.

Bali Simulation



SGI Confidential

21

Asim = verilog | C. Not defined. Transaction accurate, maybe cycle acc.

R Chip Team

Patrick Law	Scan Convertors & Texture Decompression
Chris Migdal	Texture addressing
Dave Digman	Texture Filtering & Imaging
Amy Migdal	Lighting, Texture Env, Fog
Ed Chen	Fragment Unit
David Wang	SDRAM & Network Interfaces
Allen Akin	
Chase Garfinkel	
Mahesh Dandapani	
Mark Grossman	
Phil Lacroute	
Alex Minkin	
John Montrym	
David Tannenbaum	
Lani Wells	
Rob Wheeler	
Mark Leather	(M Chip)

Home Page: http://b7.asd/bali/trees/doc/chips/r/index.html

Requirements

Area-optimized fast SRAMs

Issues

Real "N" network simulations

On-the fly texture compression

Need to simulate new fog, lighting

Multiple GE injection

External Parts

428 MHz clock generator

143 MHz 512Kx32 SGRAMs; also 1Mx16 and 4Mx16 SDRAMs

M chips :-)

sync gfx rams.

22

New Functionality

Extended range & precision color components (s10e5 float)

Per-pixel lighting

Fog is a function of Range & Elevation

2 Active Textures

Space-varying convolve

Performance

143 MHz clock (SDRAM vendor consensus)

1 pixel per clock polygon fill rate (2 tex, alpha-blend, z buf)

1 pixel per clock drawpixels

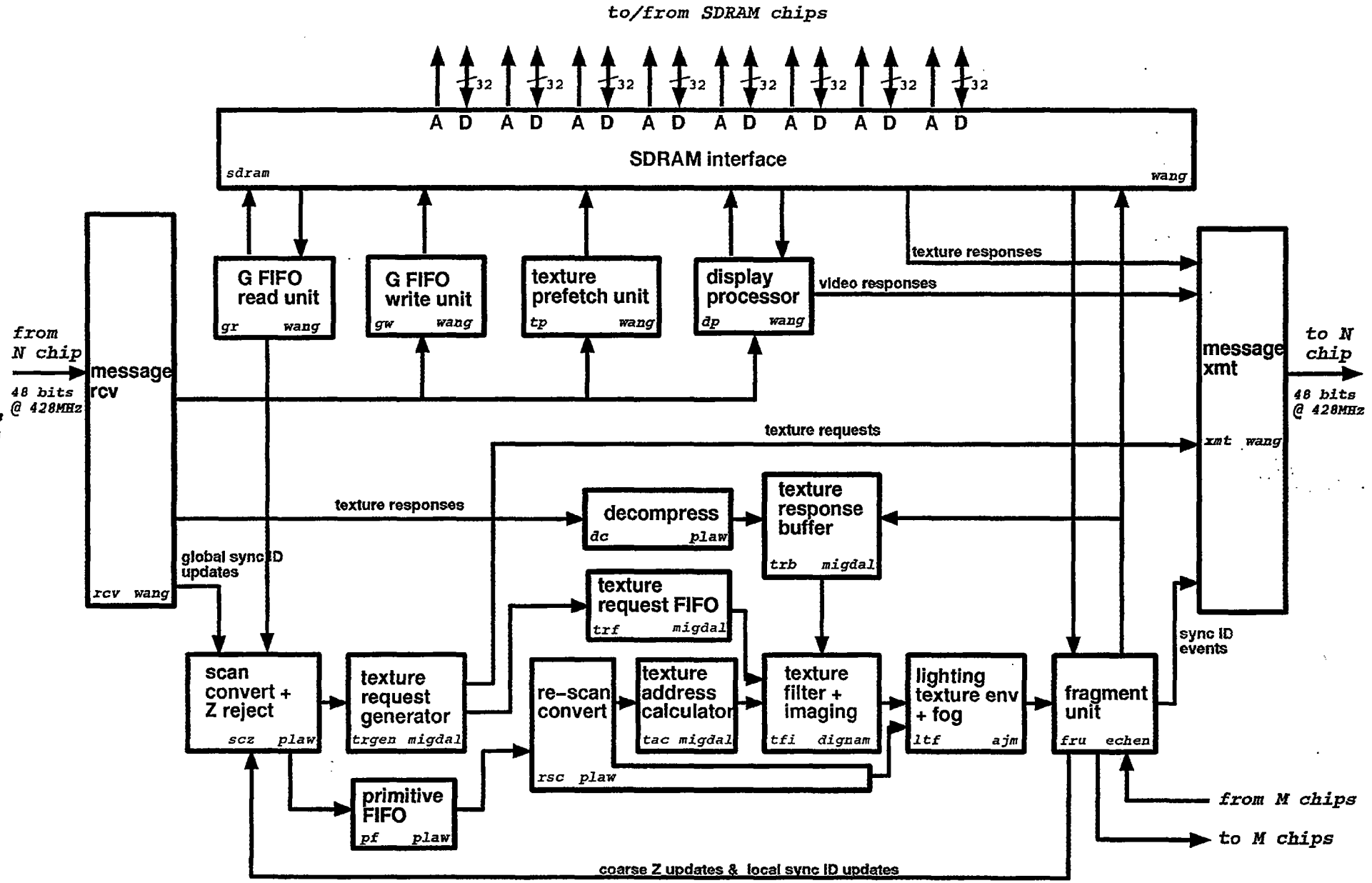
1/2 pixel per clock accumulation

1/6 pixel per clock 7x7 convolve

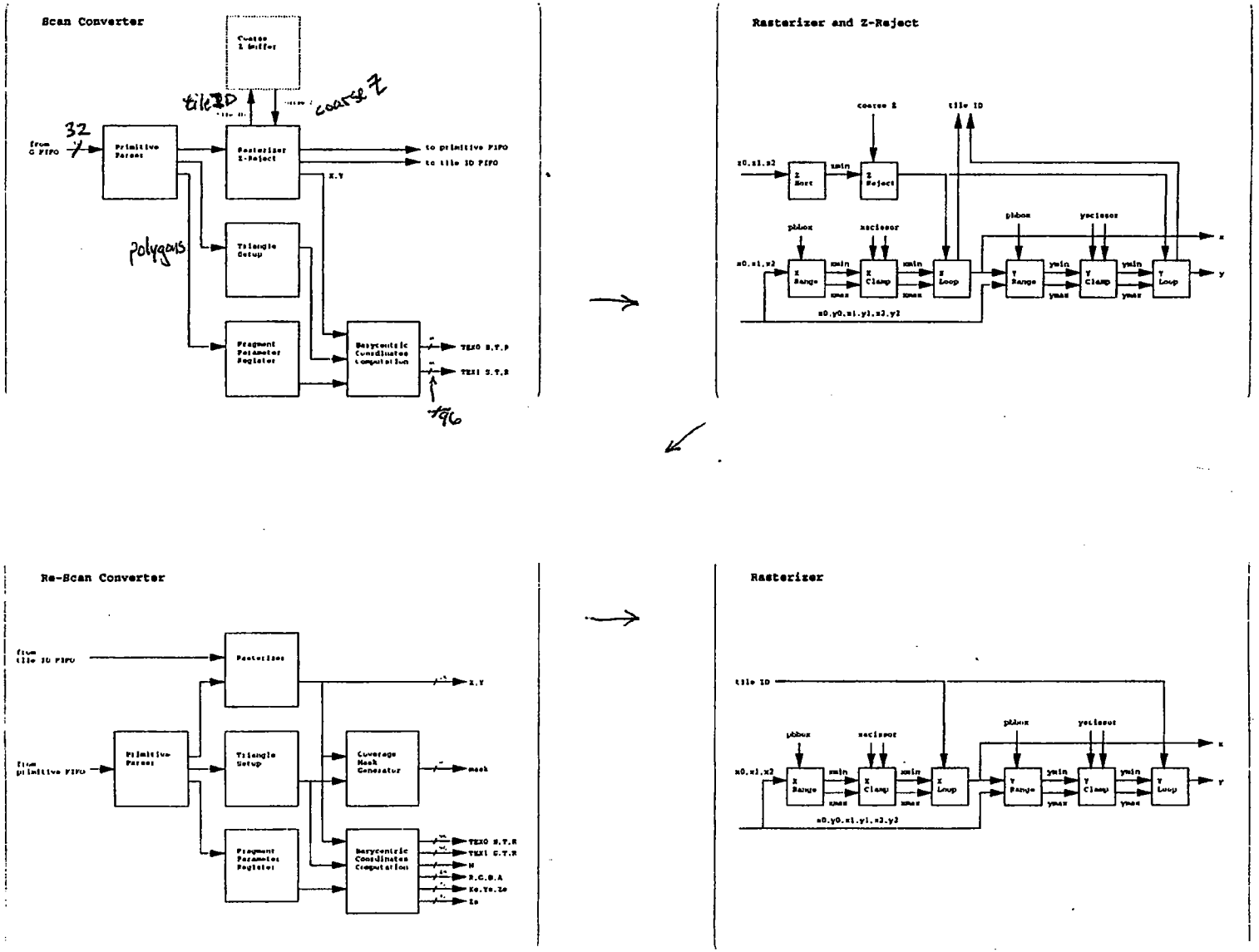
4 pixel per clock clear

64 bits per clock texture download (indep of tex format)

R Block Diagram



Patrick Law



24

Eye-space Barycentric Coordinates

f = A * f0 + B * f1 + C * f2

* color, normal, eye-space depth

$$A = \frac{a/w0}{a/w0 + b/w1 + c/w2}$$

$$B = \frac{b/w1}{a/w0 + b/w1 + c/w2}$$

C = 1 - A - B

* texture coordinates

$$A = \frac{a/w0}{q0*a/w0 + q1*b/w1 + q2*c/w2}$$

$$B = \frac{b/w1}{q0*a/w0 + q1*b/w1 + q2*c/w2}$$

$$C = \frac{c/w2}{q0*a/w0 + q1*b/w1 + q2*c/w2}$$

* screen-space depth

A = a

B = b

C = c

where

$$a = \frac{A(p,p1,p2)}{A(p0,p1,p2)}$$

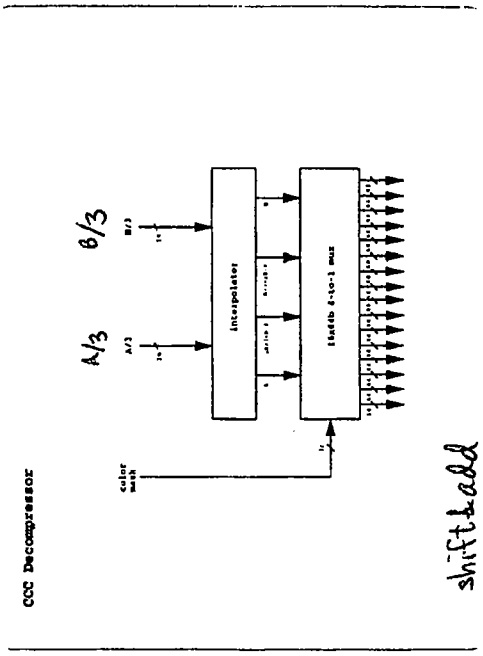
$$b = \frac{A(p,p0,p2)}{A(p0,p1,p2)}$$

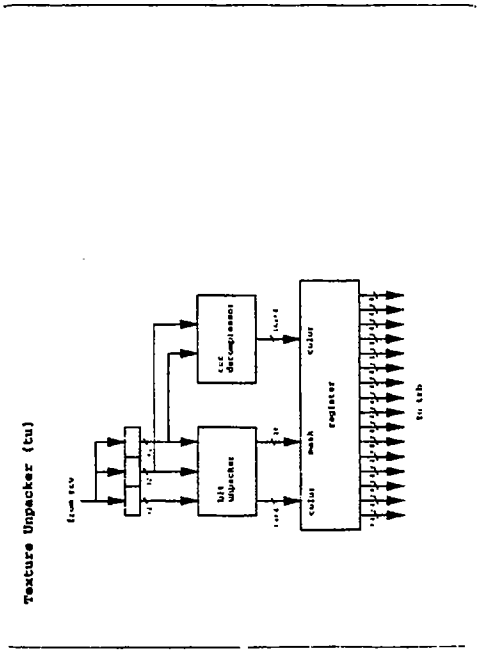
$$c = \frac{A(p,p0,p1)}{A(p0,p1,p2)}$$

assumes q sends $\frac{1}{w}$, $\frac{q}{w}$

250kg 1st scan conv.
350kg 2nd " "
May need to go to multi-pass scan conv to save gates.

SGI Confidential





Bit Unpacker

Supported Format

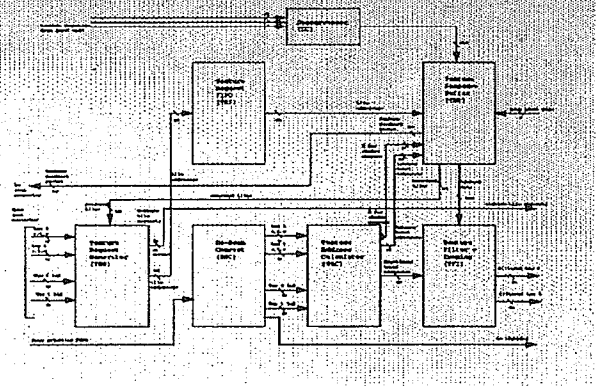
L_4N0, L_4N1, L_4N2, L_4N3	RGBA_16
LA_4L, LA_4N	RGBA_16
L_8L, L_8N	RGBA_16
LA_8	RGBA_16
L_12	RGBA_16
L_12A_4	RGBA_16
RSG6B5	RGBA_16
RGB_5A_1	RGBA_16
RGBA_4	RGBA_16
L_16	RGBA_16
LA_12	RGBA_16
RGBA_8	RGBA_16
RGB_10A_2	RGBA_16
RGB_12	RGBA_16
RGBA_12	RGBA_16
L_S10E5	RGBA_S10E5
RGB_S10E5	RGBA_S10E5
RGBA_S10E5	RGBA_S10E5
RGB_compressed	RGBA_16
RGBA_compressed	RGBA_16

Texture Subsystem

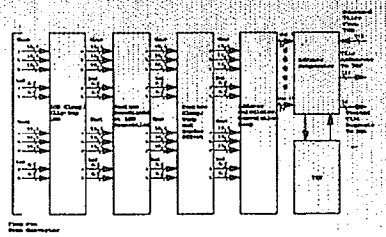
* Overview
* TRG
* TRF
* TAC
* TRB
* TFI (dignam)
* Size
* Issues

SiliconGraphics Computer Systems

Texture Subsystem Overview

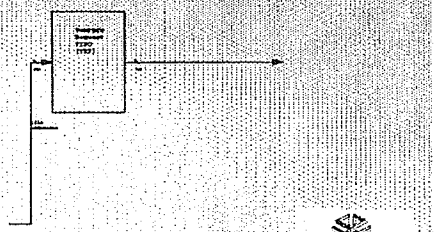


Texture Request Generator (TRG)



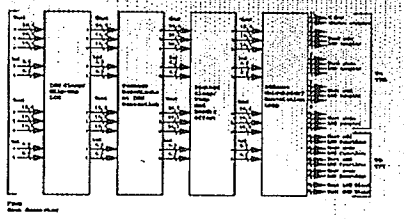
SiliconGraphics Computer Systems

Texture Response FIFO (TRF)



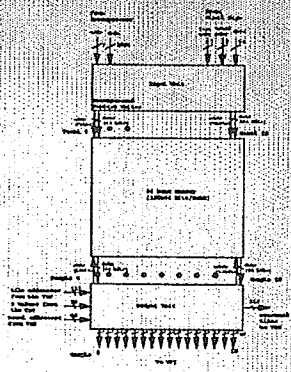
SiliconGraphics Computer Systems

Texture Address Calculator (TAC)



Silicon Graphics Computer Systems

Texture Response Buffer (TRB)



Silicon Graphics Computer Systems

Size

```
TRC
% area:          0.6% of total chip core
gates:          30.0K gates
ram:             4.5K bits
area:           2.88 sq-mm
power:           0.5  watts

TRF
% area:          1.1% of total chip core
gates:           1.2K gates
ram:            14.0K bits
area:           5.19 sq-mm
power:           0.1  watts

TAC
% area:          0.2% of total chip core
gates:          20.0K gates
ram:             0.0K bits
area:           0.76 sq-mm
power:           0.3  watts

TRB
% area:         46.4% of total chip core
gates:          11.8K gates
ram:           512.0K bits
area:          216.02 sq-mm
power:           2.0  watts
```

Silicon Graphics Computer Systems

Issues

* Size of all memories dependant on the actual latencies of the system.
* Big problem with size of current NEC memories.
* Implementation of LOD.
* Implementation of high quality textures.
* Inclusion of advanced features (3d bicubic, 3d convolutions, space varying convolves).

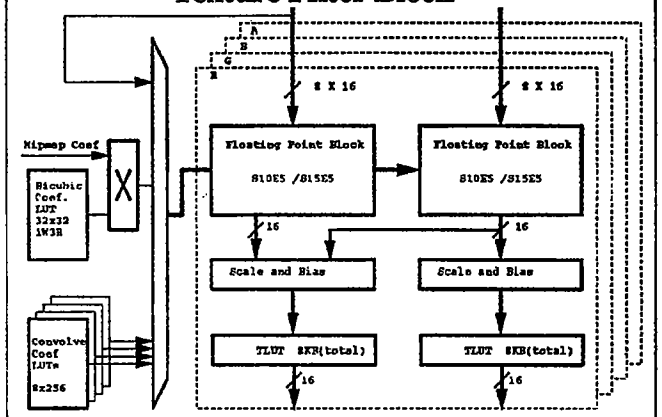
Silicon Graphics Computer Systems

Texture Filter and Imaging Features

Supports:

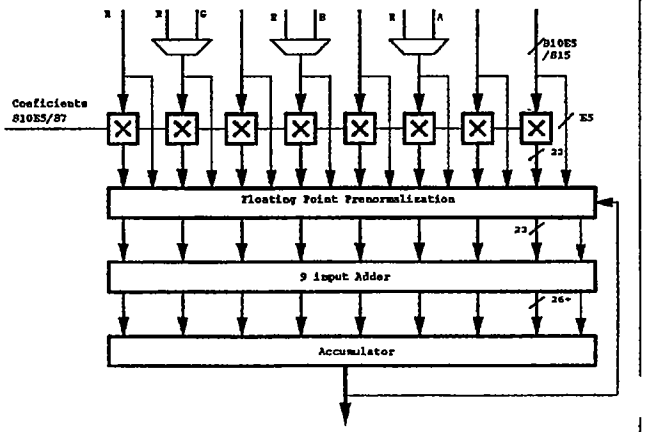
1. All the old stuff
2. Blending of two simultaneous texures
3. 2:1 anisotropic filtering
4. High quality fully anisotropic filtering (aka video filtering)
5. Floating point support for S10E5 and S15
6. Bi-cubic filtering in one pass
7. 3D bi-cubic for medical imaging
8. Coeficient table and support for 11x11 per component convoltion
9. Min/max and histogram support using tlut & tlut merging
10. Color matrix conversions
11. Space variant convolutions

Texture Filter Block



59

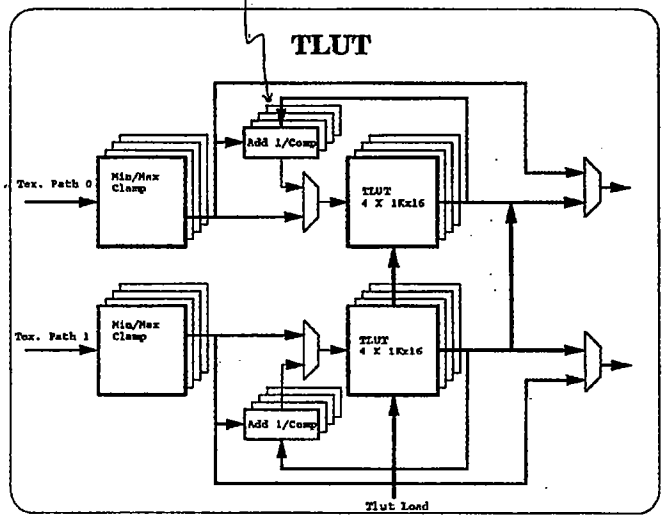
Floating Point Block



Started as a FFP tree, but had to change since need fp

For histogram support - counts hits & does min/max op.

TLUT



R Lighting

Amy Migdal
David Wang

John Airey
Avi Bleiweiss
David Blythe
Bob Drebin
Erik Lindholm
Mark Peercy
David Tannenbaum

http://b7.asd/bali/trees/doc/chips/r/lighting/lighting.html

Functionality

- Lighting
- Texture Environment
- Fog
- Alpha and Mask Logic

30

Features

- full speed (143 MPixs/sec)
- per-pixel lighting (Blinn-Phong)
- tangent space bump mapping
- environment (cube) mapping
- texture as material color or specular shininess
- two-sided lighting
- normal vector bypass
- distance attenuation
- spot light attenuation

Features, cont.

- two texture environments
- range based fog
- altitude base fog
- dithered alpha-to-mask conversion
- dithered sample mask
- alpha-to-one
- alpha function

Per-Pixel Lighting

Implemented in Lighter:

$$Lc = \frac{Att * SpotL * Cl}{Am} \{ + Dm*(N.L) + Sm*(N.H)^n \}$$

SpotL = (S.-L)>=cos(CUT_OFF_ANGLE)? (S.-L)^SPOT_EXP : 0
= 1 if disabled

Implemented in TEV:

C = Em + Am*As + S*Lc + Sm*Ey

Note: X means X can come from texture

Fog Calculation

$range = \sqrt{X^2 + Y^2 + Z^2}$

deltaAltitude = AltitudeEye - Altitude

f = exp[|FogEye-Fog| * range * 1/deltaAltitude]

Fog values are determined by indexing LUT at altitude of eye and altitude of fragment.

Texture Environment

Cv = A * Cf + B * Ct + D

Implements lighting equation

Implements TEV functions, including ADD

	A	B	D
Replace Fragment:	1	0	0
Replace Texture:	0	1	0
Modulate:	Ct	0	0
Blend:	-Ct	Cc	Cf
Decal:	-At	At	Cf
Add	1	Cc	Cb

Ct: texture color
At: texture alpha
Cf: lit fragment color
Cc: constant color
Cb: bias color

Precision

Nxyz, Lxyz, Hxyz	- S15
Colors	- {s15e5, s11e5, s11e5, s15e5}
Att	- 0.16
texture	- {s15e5, s11e5, s11e5, s15e5}
texture as normal	- clamp to S15

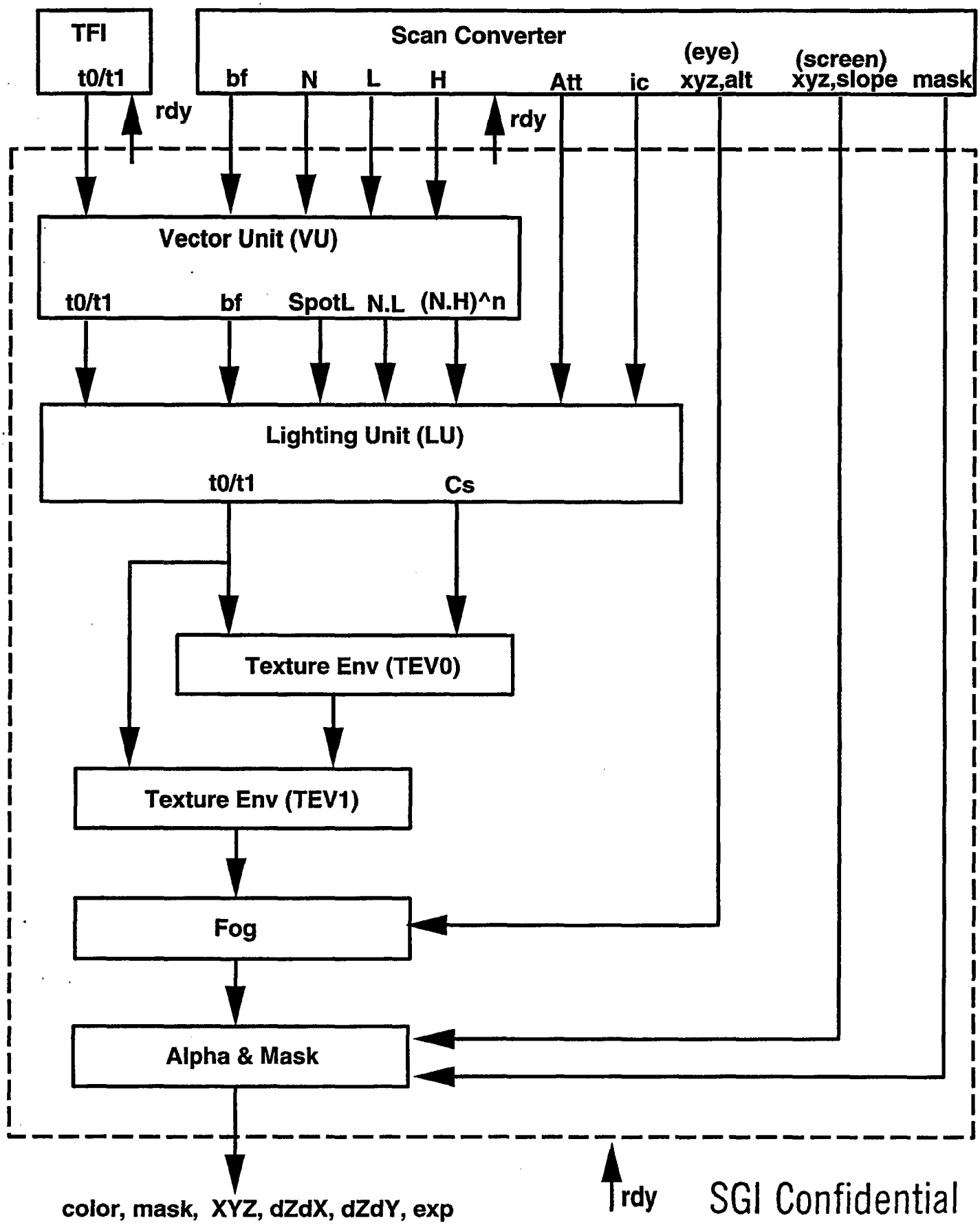
Precision, cont.

mask	- 8
X,Y,Z (screen)	- (12, 12, s31)
dZdX, dZdY	- s15
exp	- 5
X,Y,Z (eye)	- s23
altitude (eye)	- 12

Issues

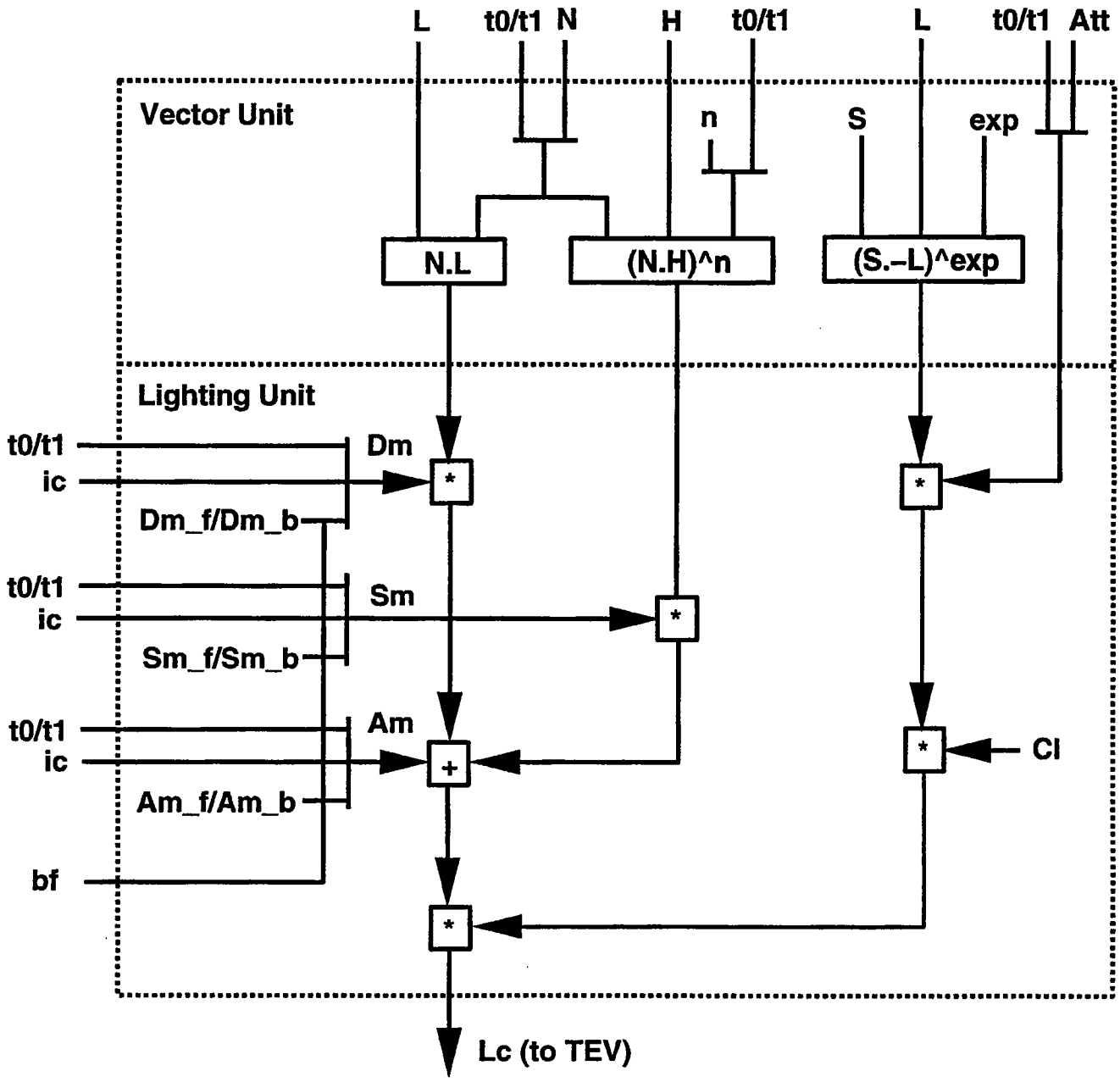
- Spot light aliasing - linear fall-off?
- pre-lighting TEV?
- Interpolated Att OK?
- Approximate range for fog OK?
- Z-based fog also necessary?
- Can per-pixel lighting be used instead of dedicated fog hardware?

LTF Block Diagram



SGI Confidential

Lighter Block Diagram

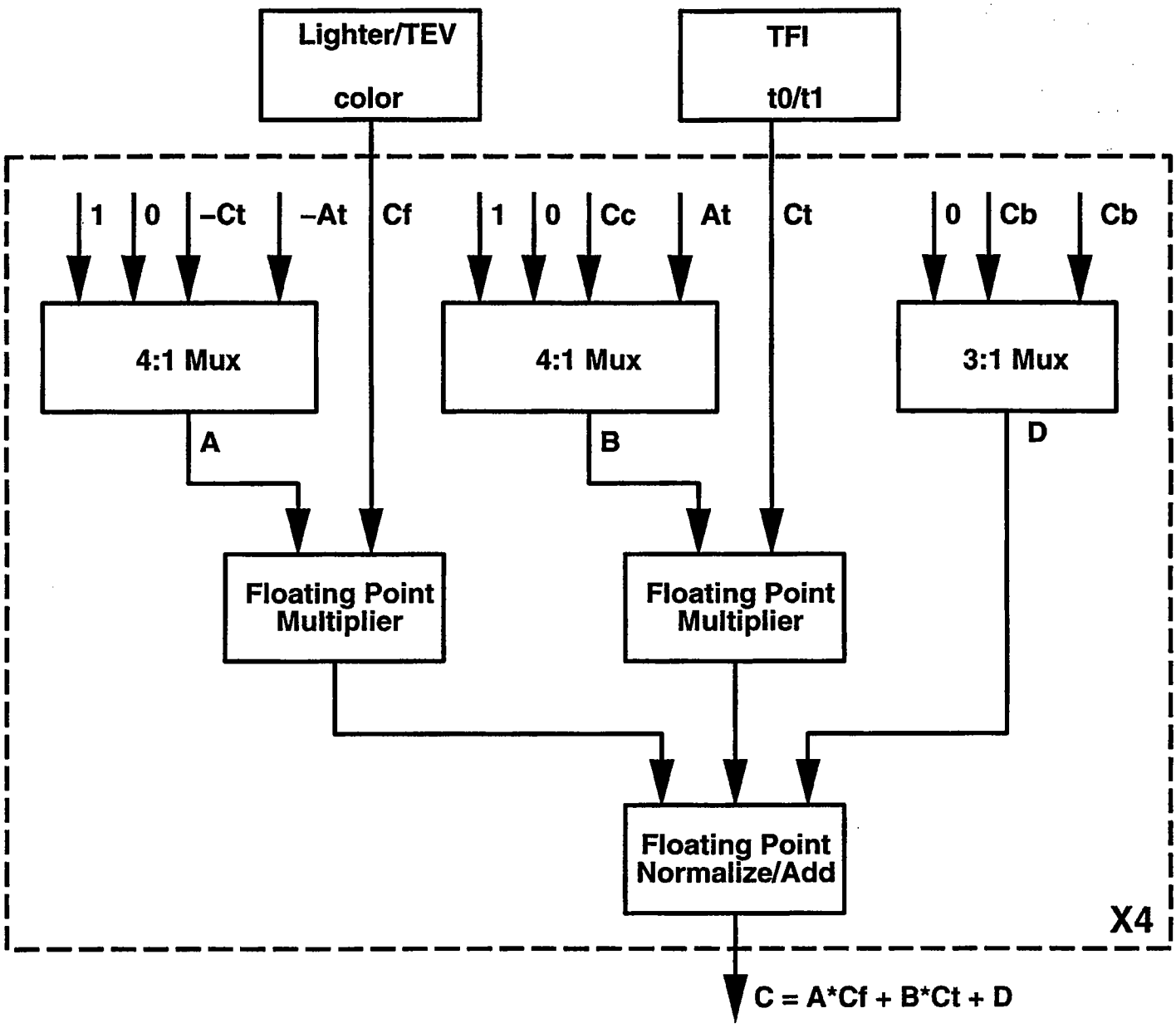



```
Lc = Att*SpotL*Cl  {      // distance*spot light attenuation*light color
          Am              // ambient material*ambient light
        + Dm*(N.L)        // diffuse material*diffuse light
        + Sm*(N.H)^n      // specular material*specular light
     }
```

ic = interpolated color
t0/t1= textures
*_f=front material color
*_b=back material color

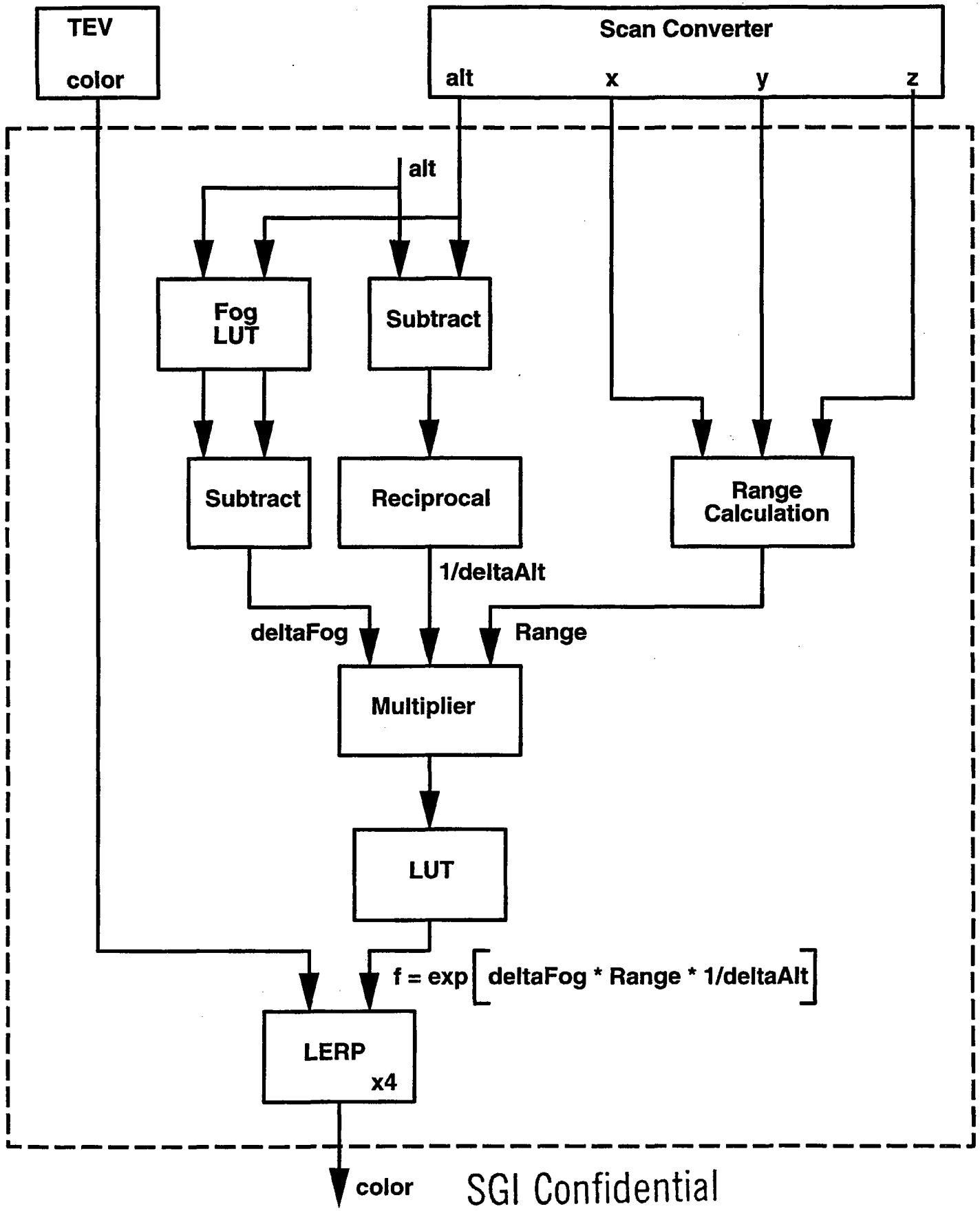
SGI Confidential

Texture Environment Block Diagram

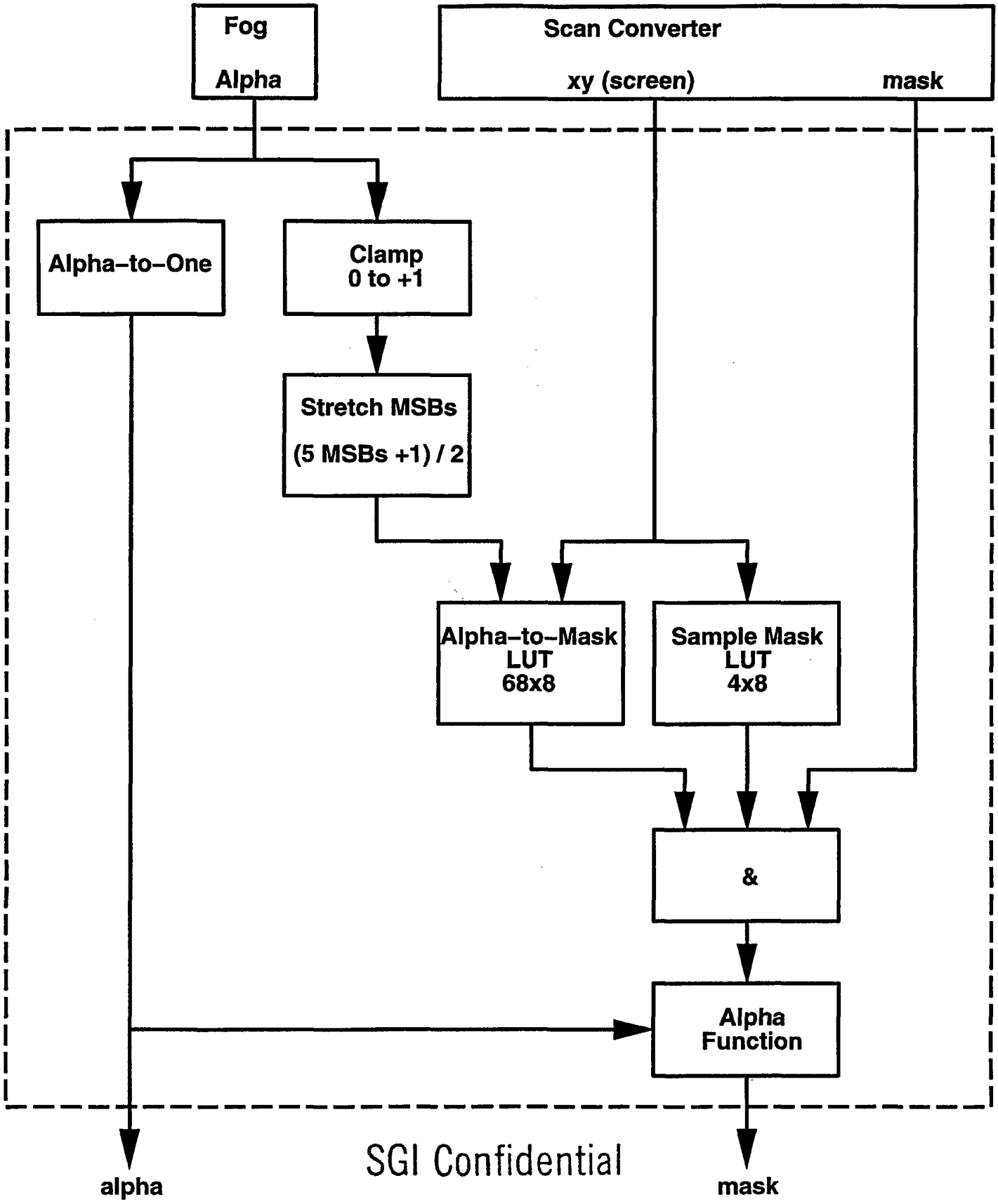


Ct: texture color	Replace Fragment:	A = 1	B = 0	D = 0
At: texture alpha	Replace Texture:	A = 0	B = 1	D = 0
Cf: lit fragment color	Modulate:	A = Ct	B = 0	D = 0
Cc: constant color	Blend:	A =-Ct	B = Cc	D = Cf
Cb: bias color	Decal:	A =-At	B = At	D = Cf
	Add	A = 1	B = Cc	D = Cb

Fog Block Diagram



Alpha and Mask Block Diagram



FRU Functions

* Stencil, depth, blend, logic–op non–multisampled pixels
* Supported fb formats:
  S10E5, RGBA12, LA16, CI16
* Accumulation buffer
* Fast clear
* Pass fragment info to M chips, receive resolved color from M
* Automatic & explicit Z cull
* Calligraphics
* In–place copy & blend
* Partial resolve for multipass shading

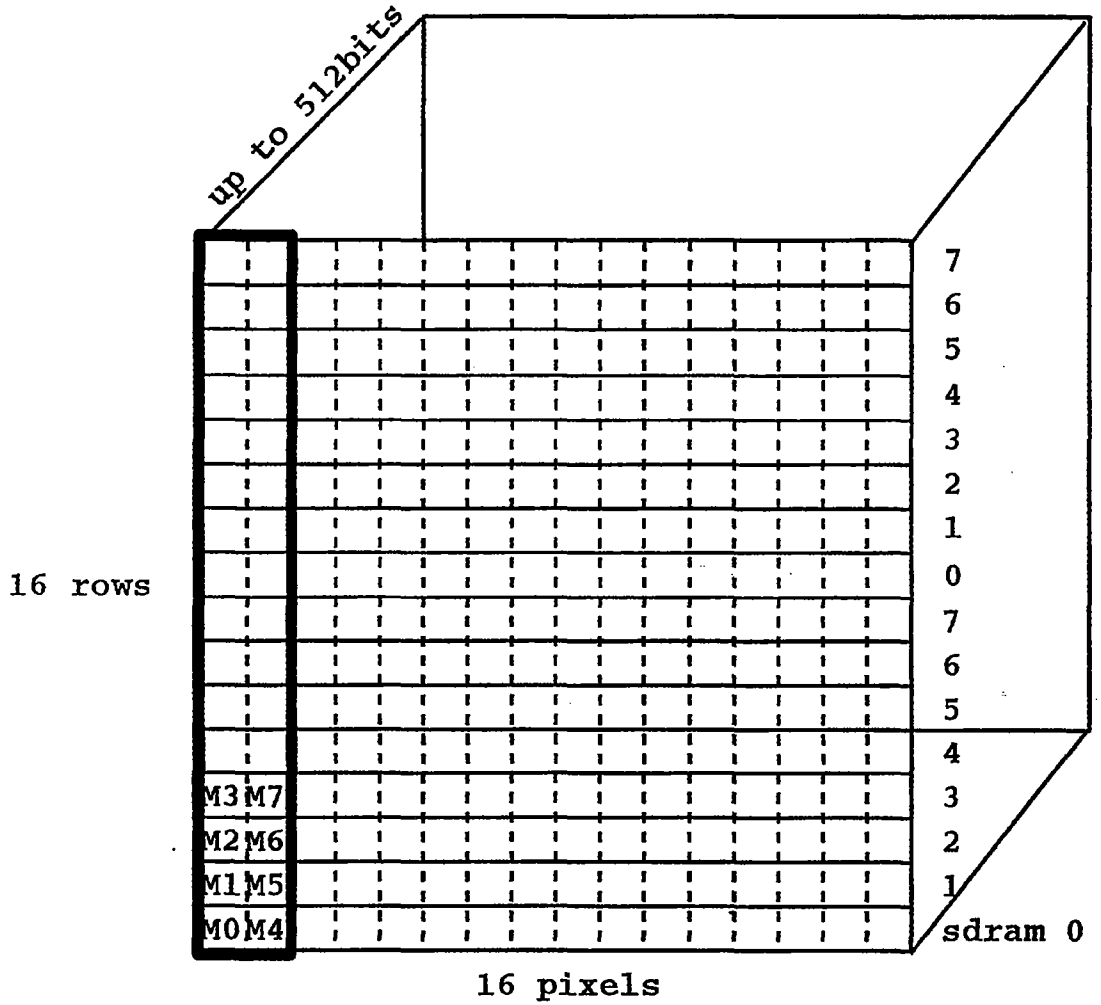
FRU Performance

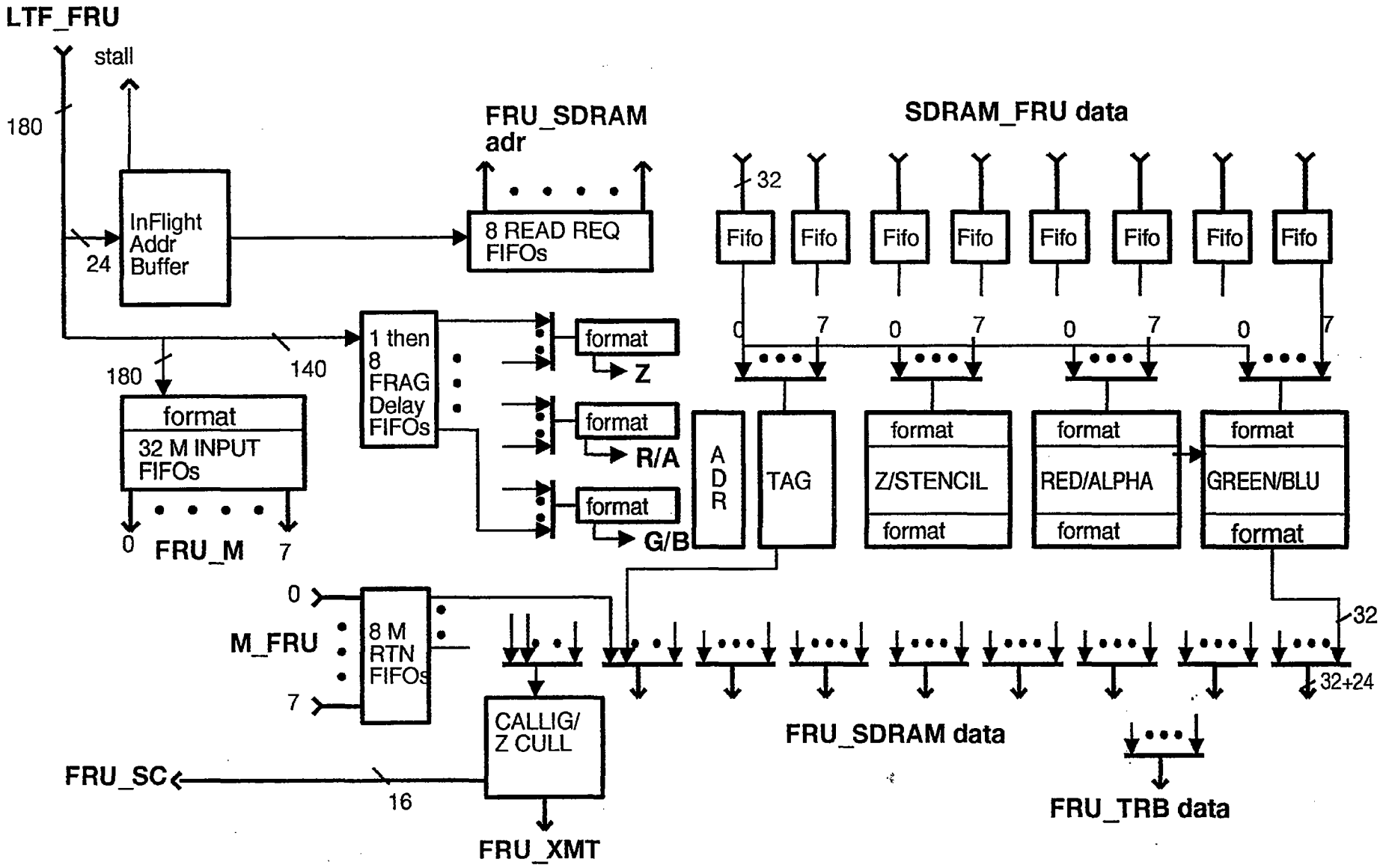
* One depth/stencil tested, blended pixel per clock
* One accum buf operation every 2 clocks
* Fragment packet from ltf = 180 bits

* Desired 1 pixel/clock performance = 143 MPixels/sec
* Pixel = (RGBA, SZ) = 16*4 + 32 = 96 bits
* RW of a pixel = 192 bits/clock = 3400MB/sec peak

FRU sdram Memory Map

Rev 1 11/11/96





FRU Block Diagram

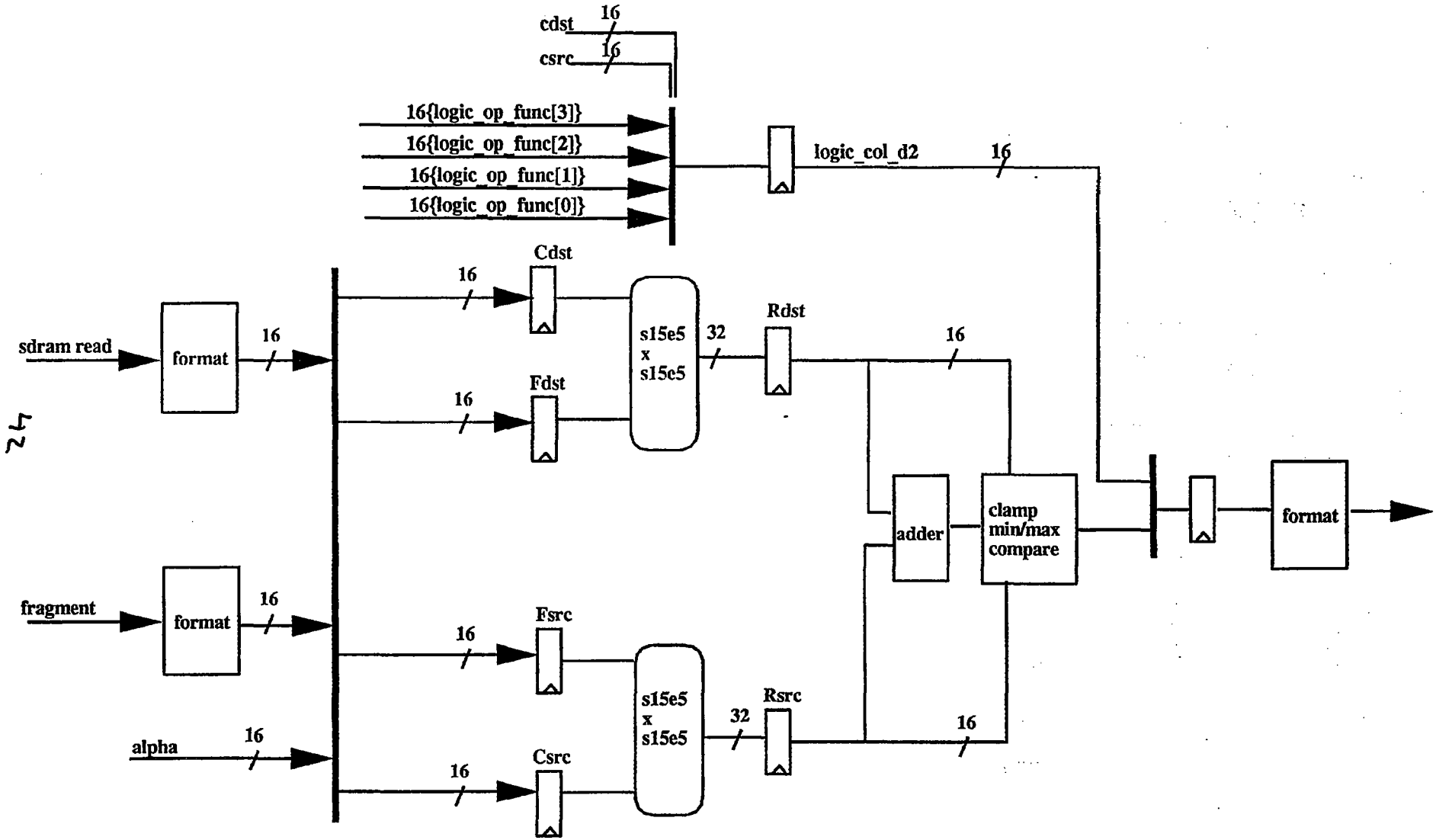
Rev 1.2

SGI Confidential

41

FRU Blend

Rev 1 11/11/96



42

FRU Issues

* Tiling & load balancing
* Better solution for pixels–in–flight problem?
* FRU_TRB data
* Final supported pixel formats/conversion
* Additional blend functions?
* Alternative implementations

43

R SDRAM & Network Interfaces

David Wang
John Montrym

http://b7.asd/bali/trees/doc/chips/r/rinf_offsite2.sc

Vital Statistics

8 32-bit wide SDRAMs

BW = 143MHz * 32bytes * 80% = 3660 MB/sec

- 360MB/sec in — G FIFO write
- 360MB/sec out — G FIFO read
- 70MB/sec out — video refresh (1280x1024x72Hzx6bytes, 8R)
- -1054MB/sec out — texture response (limited by chip I/O)
- -1816MB/sec in/out framebuffer

(ref: doc/chips/r/bw.txt)

Memory size per R = 16MB (use 512Kx32 SGRAM)
32MB (use 2*1Mx16 SDRAM)
128MB (use 2x4Mx16 SDRAM)

Design Goal

Command queue based interface allows fire-and-forget protocol

Dedicated queues for texture and framebuffer access

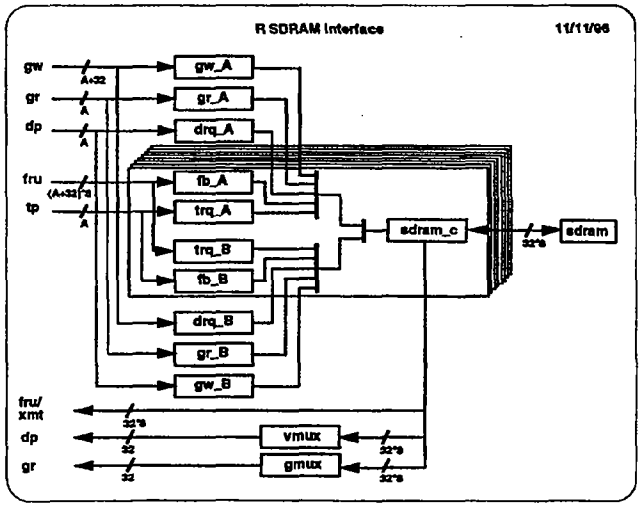
Separate queues for SDRAM bank A and B to hide page miss latency

256-bit atom access allows zero overhead bank switch

Round-robin arbitration with flow control

Display request has the highest priority

Display request can lock the current SDRAM to allow random access within the same page



R SDRAM Interface

Issues

Max display response latency

Burst size = 2 to allow issuing new page activate command in the middle of transfer?

Bus bandwidth

- framebuffer (A+32)*8/32*8
- display processor A/32
- G FIFO read A/32
- G FIFO write A+32
- texture prefetch A/32*8

Support 4-bank SDRAM?

Performance optimized arbitration

Bali Offsite 2

R Network Interface

Vital Statistics

Three 16 bit @ 143MHz*3 input and output channels

BW = 143MHz*3*2bytes*3 ports*75% = 1930MB/sec

R chip input budget
- 360MB/sec in triangle FIFO
- 9MB/sec in video requests
  (rough: 1280x1024x72Hz 8 R system)
- leave 1561MB/sec for texture req & responses
  assume 15% tex req overhead
  -> 1170 MB/sec for texture data (responses)

R chip output budget

- 48MB/sec out video responses
  (1280x1024x72Hzx4bytes, 8R)
- 1561MB/sec out texture req & responses

Bali Offsite 2

45

R Network Interface

Design Goal - rcv

Match or provide higher BW than Bali net

Individual flow control for
- texture response packet
- display request packet
- G FIFO write packet

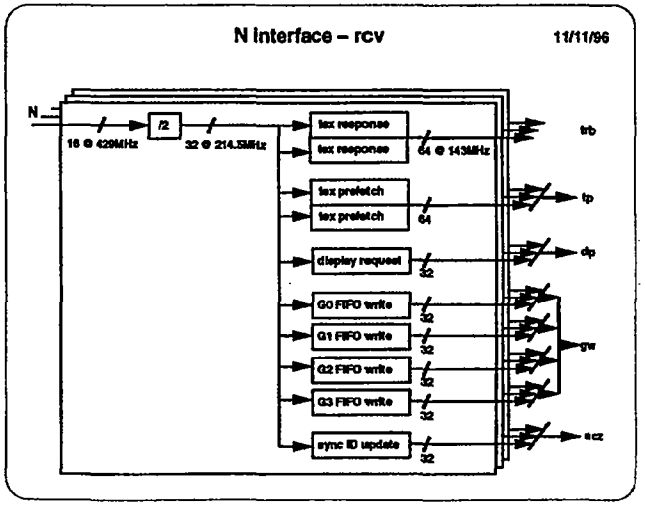
Throttle-back message for G->R flow control

Round robin among channels

Need to re-sort packets from a specific G

Half tile response per (R) cycle
- we can achieve ~0.7 tile per cycle

Bali Offsite 2



45

R Network Interface

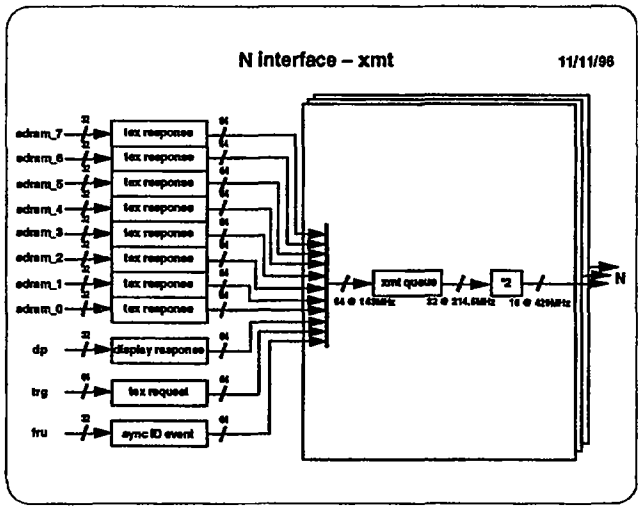
Design Goal - xmt

Display response has the highest priority

Round robin arbitration for others

Half texture request per (R) cycle
- we can achieve one texture request per cycle

Bali Offsite 2



R Network Interface

Gate Count

** sdram **	
% area:	2.7% of total chip core
gates:	290.0K gates
ram:	0.0K bits
area:	12.35 sq-mm
** xmt **	
% area:	4.5% of total chip core
gates:	261.1K gates
ram:	24.0K bits
area:	21.11 sq-mm
** rcv **	
% area:	2.5% of total chip core
gates:	279.0K gates
ram:	0.0K bits
area:	11.62 sq-mm

Bali Offsite 2

R Network Interface

Issues

Three narrow N channels overhead

Throttle-back flow control scheme

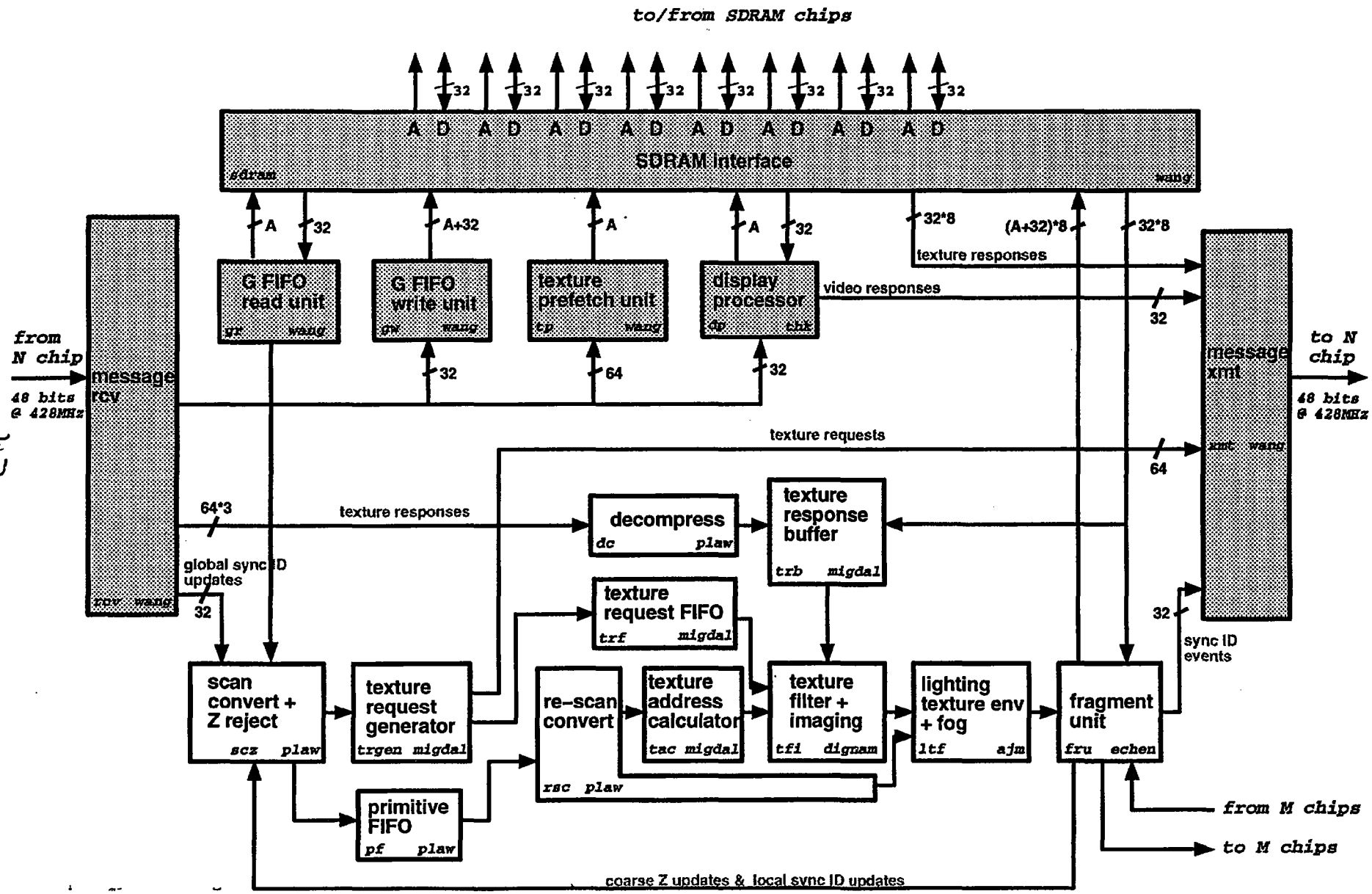
Force a given G to use specific channel?
avoid sorting G packets

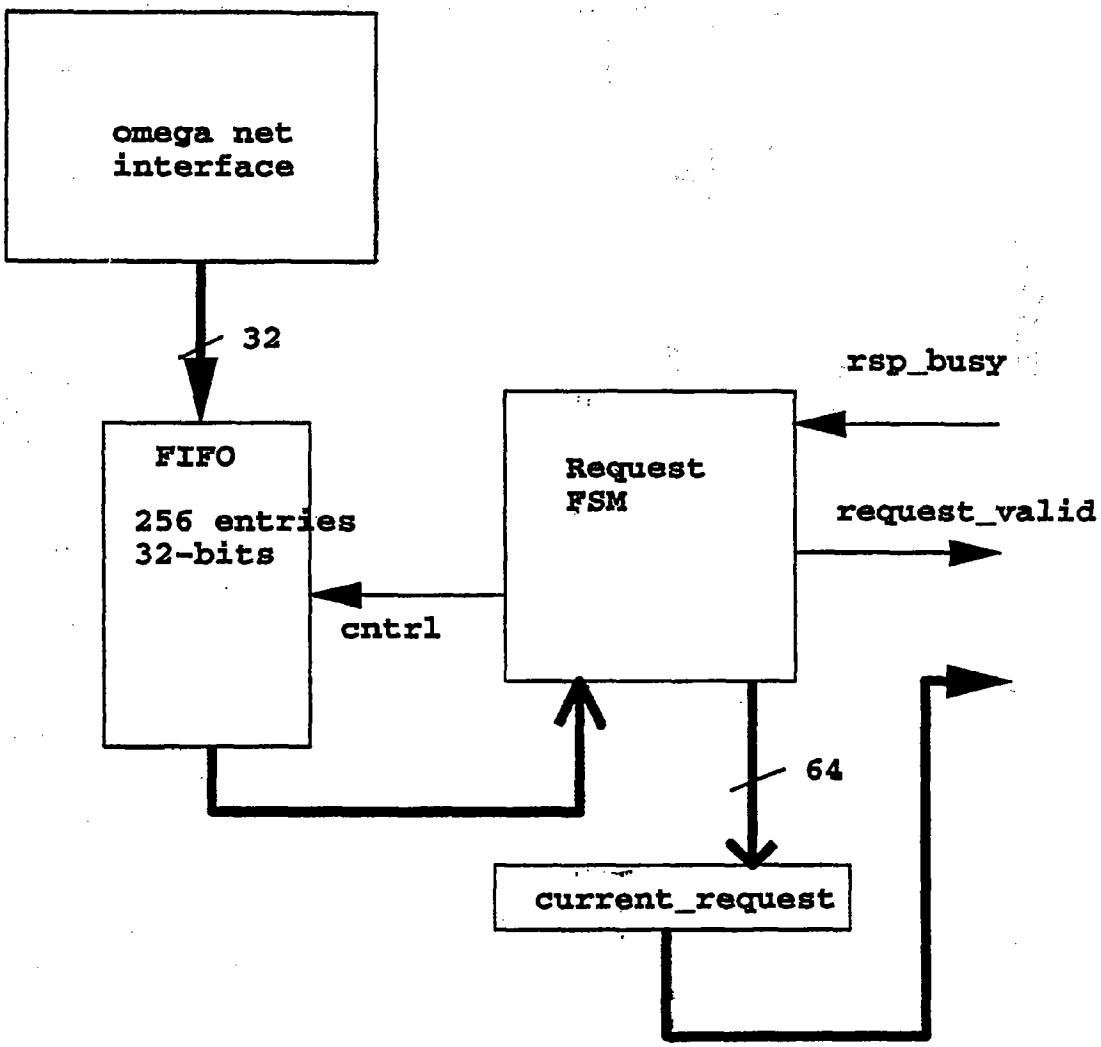
Error detection/correction

Need high speed (214.5MHz) SRAM for N write

Bali Offsite 2

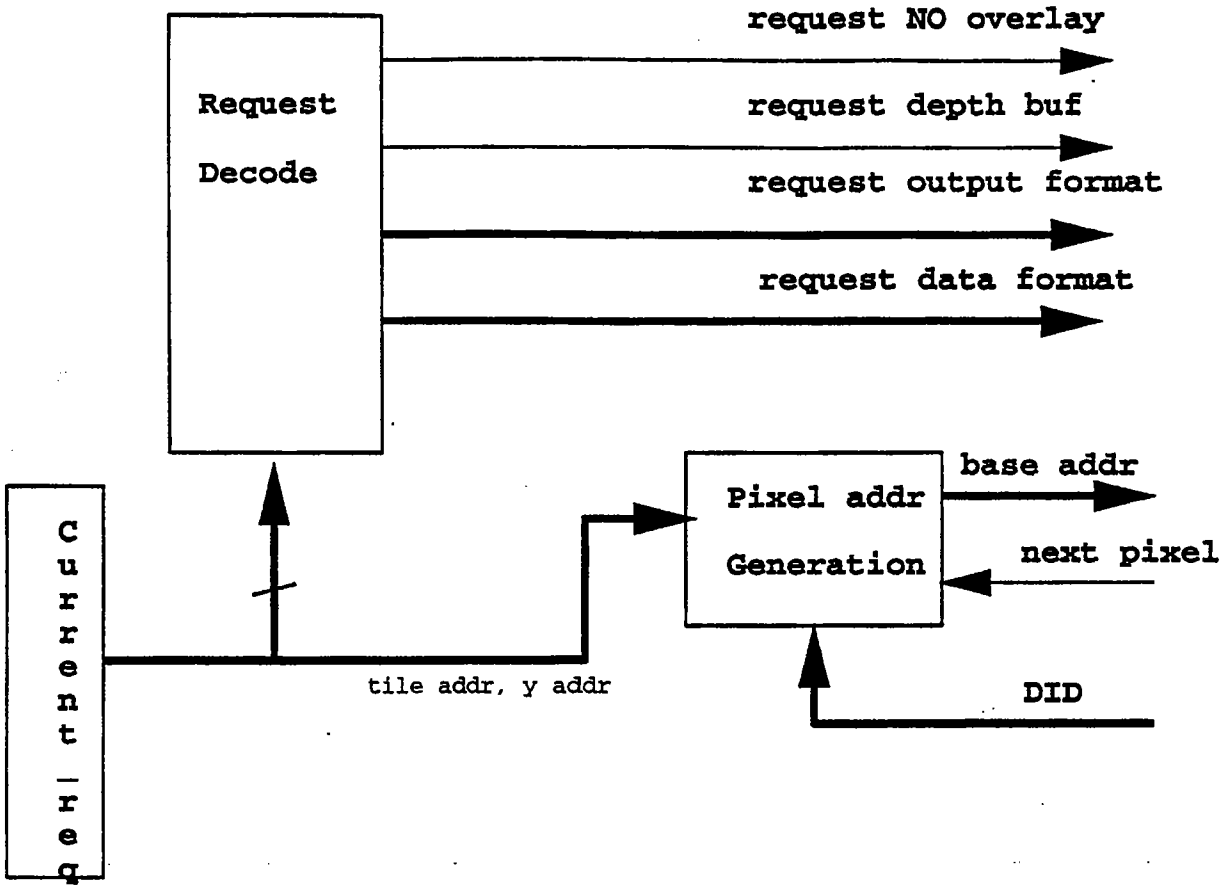
R Block Diagram – SDRAM centric

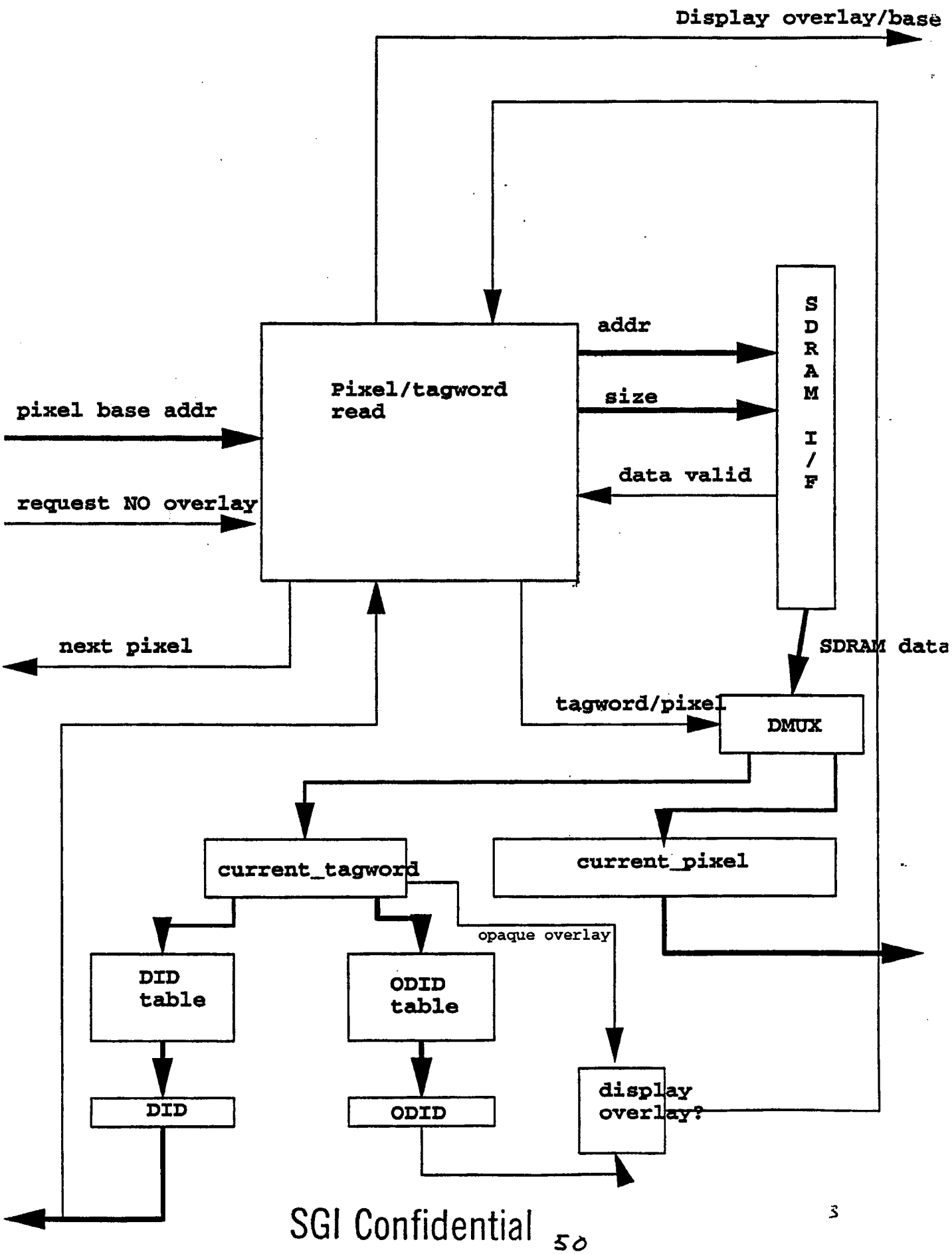




RAM: 256 x 32 bits for FIFO
FFLOPS: 64 current_request
9 fifo rd pointer
9 fifo wr pointer
1 state bit for FSM
1 request_valid
1 rsp_busy

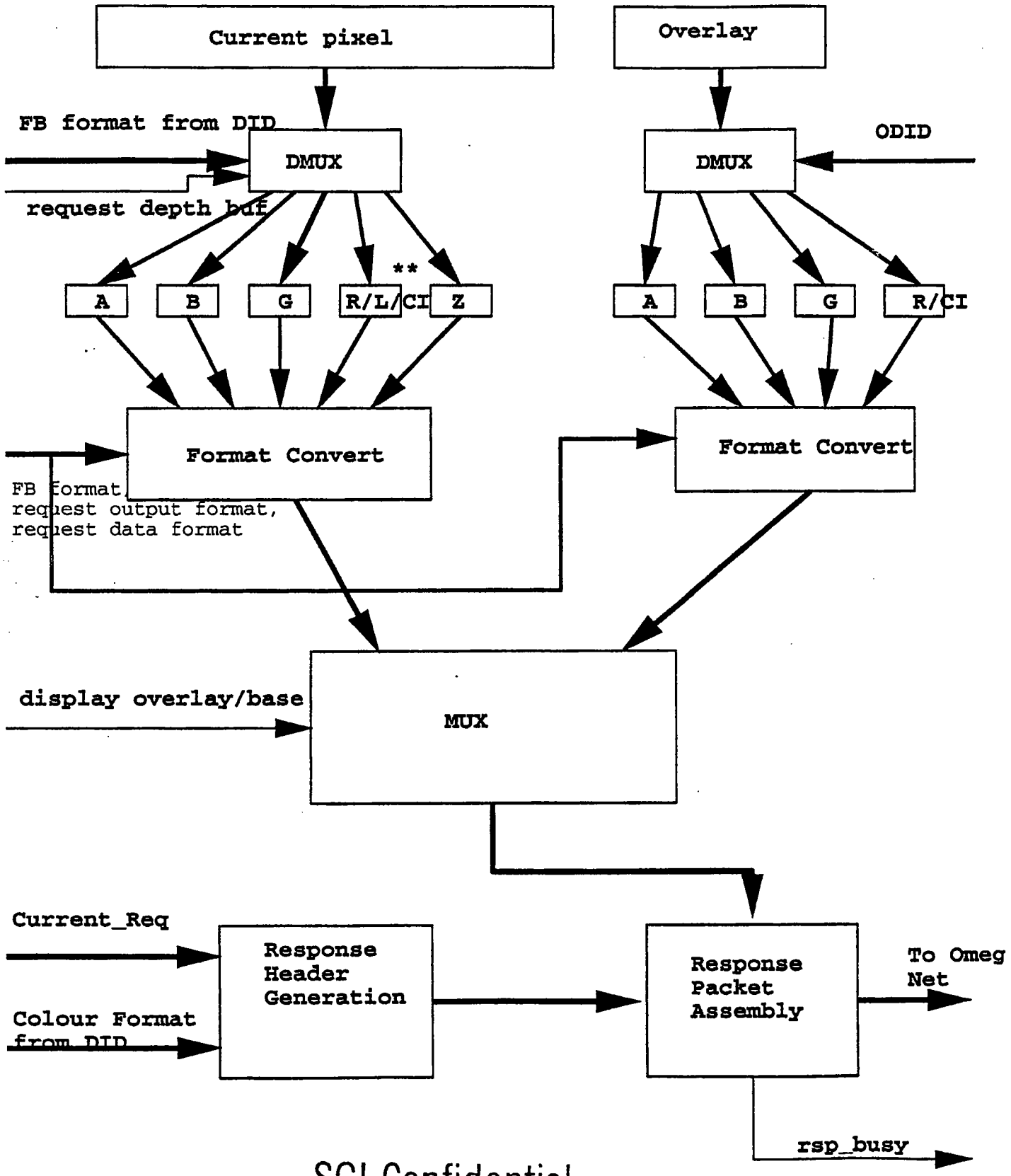
Clock count: 1 to get to front of FIFO,
1 to get out of FIFO into Request FSM
2 in the FSM.





SGI Confidential 50

3



SGI Confidential

51

4

Overview

Scalar core programming environment
Pixel operation performance
- Throughput
- Setup

Open issues
- Scalar Core
- Pixel path

Scalar core

We are assuming:
- 200 MHz performance
- RISC instruction set
- Fast immediates & bitwise ops
- Floating point support C compiler
- Simulator w/ interactive debugger

Register writes
- Immediate to G chip: 1 cycle
- Immediate + mask to PE: 2 cycles

Fast interface to G & vector units
- What if the core is external?
  - G chip register reads
  - Round-trips / semaphore checks

Low entry-point overhead

→ arm used to save unnecessary caller state to stack.

Pixel Peak: Simple stuff

(Assuming DUPLO interface) → New Lego Orange #'s.

RGBA8 DrawPixels	**325 MPix/s**
RGBA8 ReadPixels	**283 MPix/s**
RGBA12 copy	
In-place	1000 MPix/s
Not in-place	536 MPix/s

Pixel Peak: Useful stuff

(Assuming DUPLO interface, 8 R's)

RGBA12 copy	
5x5 convolve	350 MPix/s
11x11 convolve	121 Mpix/s

Pixel Setup: GI & GE

Write from host

IDMA Engine:	~20 cycles
Address, width, skip	
Debubbler:	~250 cycles
Dead bytes for scan lines	
Converter:	~80 cycles
Input/output type	
Command table:	~100 cycles
Total:	**450 cycles**

Read to host

Fetch unit:	~150 cycles
SDRAM addr, x, y, width, height	
Convertor:	~80 cycles
Source type, dest type	
ODMA Engine:	???

Pixel Setup: PE

Write from host:

Input/output types	
SDRAM addr	
x, y, width	
Reload from scalar:	~150 cycles

Copy & Read ops:

Nothing?	0 cycles?

Pixel Setup: R Chip to TFI

Promoter	**~8 registers**
Incoming format	
TRG	**~8 registers**
Texture filter mode	
(convolution / color matrix)	
TFI Filter mode	**~4 registers**
TFI Multiply/accum	
121*4 coefficients (max)	242 words
4*4 coefficients (matrix)	8 word

Pixel Setup: LTF & IMP

LTF Pre-LUT scale/bias	**8 registers**
LTF LUT/histogram	
Configure	8 registers
Load from SDRAM	???
LTF Post-LUT scale/bias	**8 registers**
8 PE shadow reg writes	~16 cycles
IMP Convertor	**4 registers**
IMP Color Mask	**3 registers**
IMP Blending Mode	**1 register**

Pixel Setup: R Chip Total

Total: **51 registers**
Excluding convolve coefficients
If written from scalar:
Writes alone: ~100 cycles
Get from EMEM: ???
Compute ???
Should happen on mode change

Pixel Setup: Misc

DrawPixels overhead
Lazy mode stuff etc: ~180 cycles
Draw textured rectangle
Send directly to PE ~200 cycles
Restore G state
< 8 register writes 16 cycles
Restore PE state
Polymode etc. 16 cycles?
Restore R state
Is this necessary every time?
2 copies of shadow on PE?
Other lazy validation scheme?
Leave PE shadow in geometry state?
Pixel context switching expensive

54

Pixel Setup: Useful draws

DrawPixels (no transfer modes)

G chip setup:	700 cycles
R chip setup:	800 cycles
State restore:	??? cycles
Total:	**1.5K cycles**
Ops/sec:	**133 K**

DrawPixels (convolve)

G chip:	700 cycles
R chip (download):	800 cycles
R chip (draw):	800 cycles
Draw rectangle:	200 cycles
State restore:	??? cycles
Total:	**2.5K cycles**
Ops/sec:	**80 K**

Pixel Setup: Useful copies

CopyPixels w/ LUT & scale/bias

Copy to scratch:	200 cycles
Configure R chip:	800 cycles
Draw rectangle:	200 cycles
Total:	**1.2K cycles**
Ops/sec:	**166 K**

CopyPixels w/ small convolution:

Copy to scratch:	200 cycles
Configure R chip:	1000 cycles
Draw rectangle:	200 cycles
Total:	**1.4K cycles**
Ops/sec:	**140 K**

CopyPixels w/ LUT & convolve

Configure R chip (twice):	1800 cycles
Draw two rectangles:	400cycles
Total:	**2.2K cycles**
Ops/sec:	**90 K**

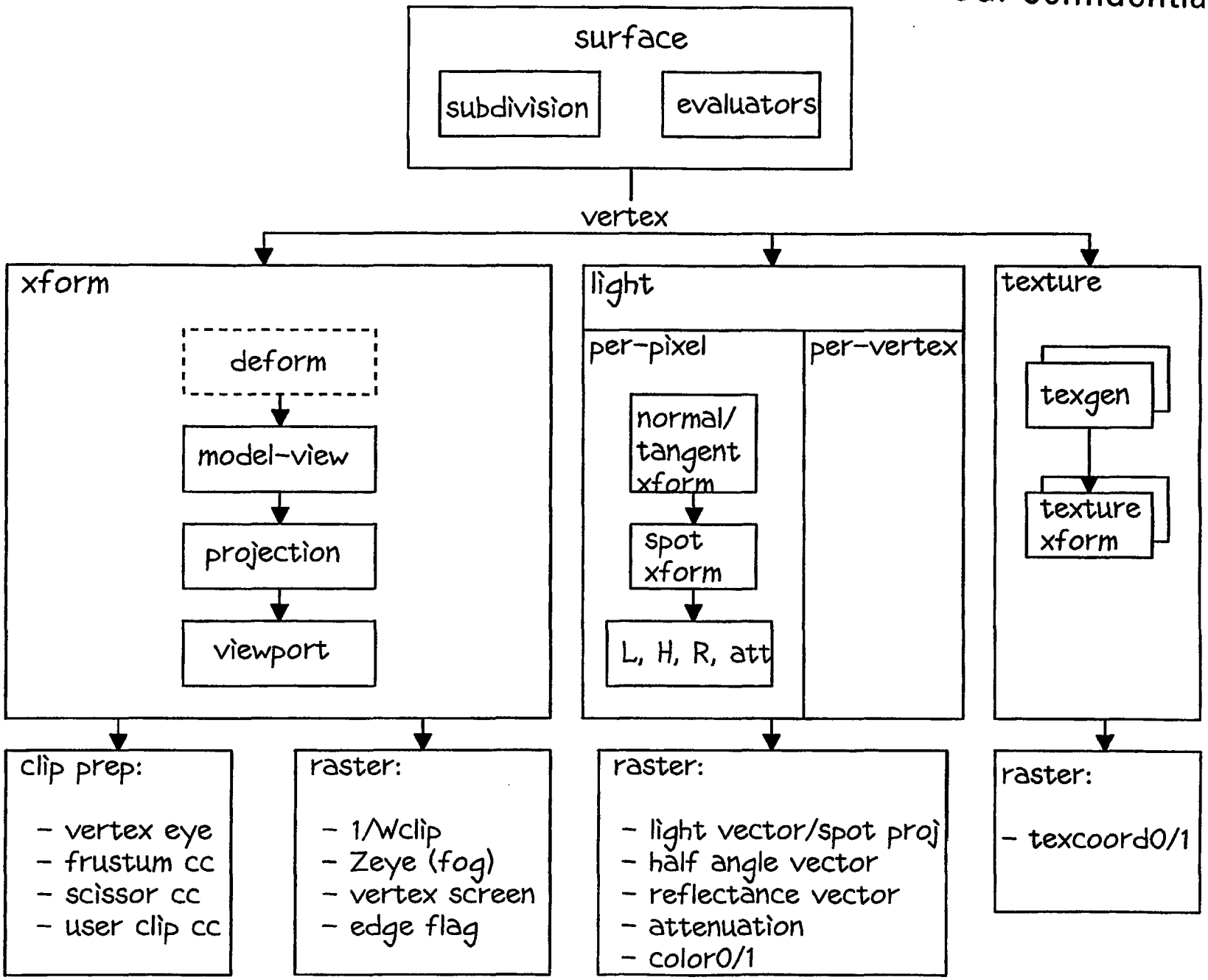
Biggest Unresolved Issues

Scalar core
- Are our expectations realistic?
- What if it's external?
- Code space / cache requirements

Pixel Path
- More specifics on setup
  - Software design
  - R chip control / cycle cnts
  - HW state management
  - Mode change speeds
- What can we simplify?
  - Pixel textures
  - Multiply/accumulate
  - Multiple spigots

55

Geometry Path



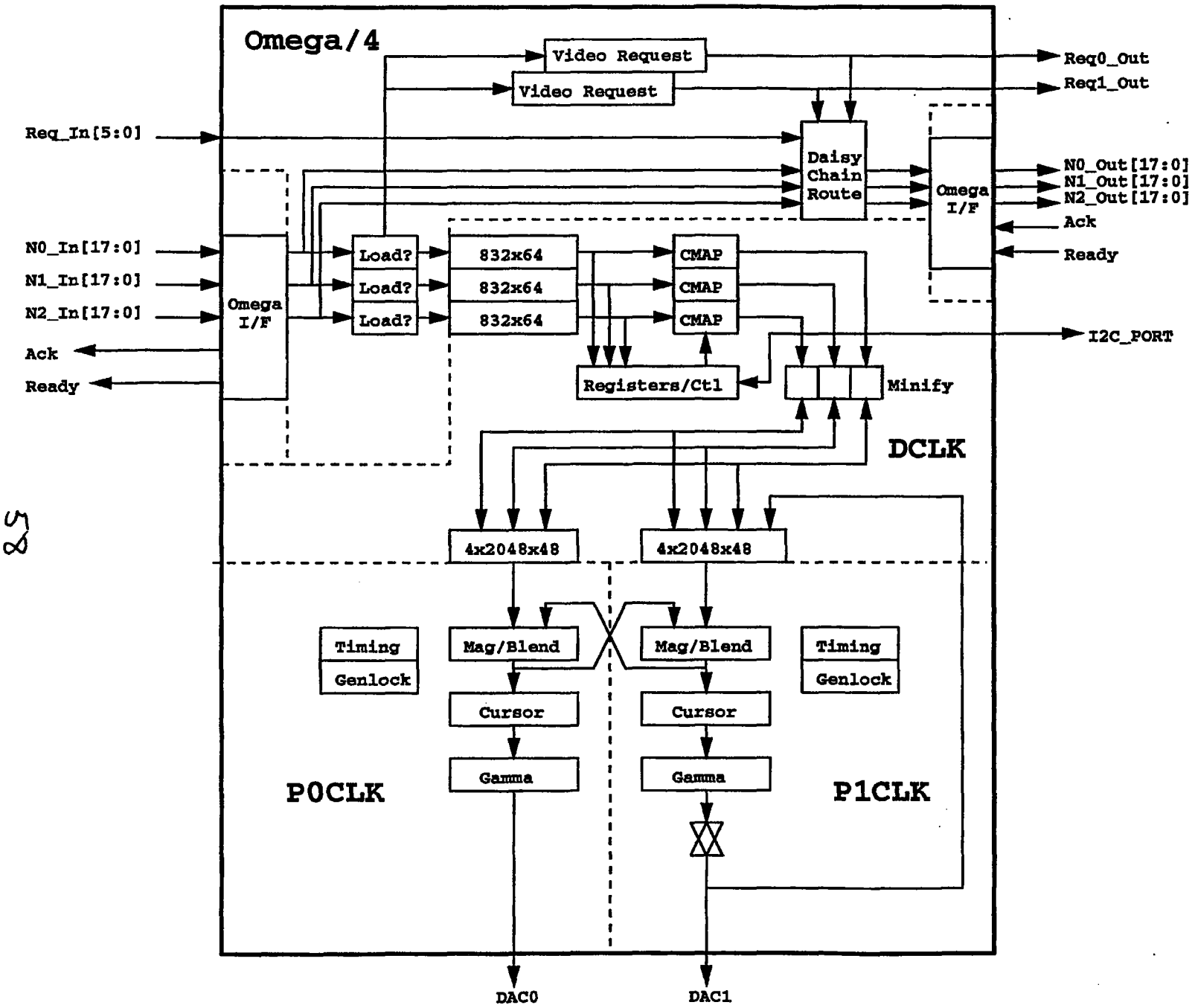
Geometry Path (cnt'd)

- What's new:
  - Better surface to vertex integration
  - Per-pixel lighting support
  - No overhead primitive switch
  - Parallel matrix operations
  - Double buffered xform state
  - Multiple projection and viewport definitions

Software Development Environment

- G software simulator for ucode development
- Vector code scheduler, assembler (back end)
- Scalar and vector code separated, shared headers

57



Design Issues

Pixel Ordering
- Up to 512 outstanding requests (4K pixels)
- 3 responses returning simultaneously
- No ordering in response sequence

Maximum number of D Chips in a system
- 32 Channels max.
- 16 D Chips provide 32 channels
  - SVGA only (15-pin din connectors)
- 8 D Chips provide 16 channels of hires output
  - Possibly a 64R system could have 32 hires outputs on 8 taps.

Latency Issues
- What is the maximum number of requests that are in line ahead of your request?
- In a 32-channel system, clearly 31 requests could be ahead of yours.
- Assuming an access for tag, overlay, and three for color - 80 accesses/packet
- 2560 memory accesses; 3:1 clock ratio to omega;
  7680 omega clocks + latency = 8000 omega clocks
- The one tough channel could be 1280x1024 (100MHz)
  8000 omega clock = 2000 pixels @ 100MHz.
- Therefore we need to prefetch about 2000 pixels ahead and account for them all coming back in one burst.

Design Issues (cont.)

Large Format Projectors (2560x2048) require pixel rates of 500MHz
- Transfer BW of 405Mpix/sec

Input Formats:
- 16-bit per channel (1-3 channels per component) 616/343/242 Mpix
  - Unsigned 16-bit
  - Signed 15-bit (for conversion to YUV)
- 12-bit per channel (2 or 4 channels per component) 440/242 Mpix
  - Unsigned 12-bit
  - Signed 11-bit

Calligraphics:
- Push-mode response packet
  - D thinks its in vampire mode
  - Just pass the data to DVO

Color Map
- 1 4K Map
- 4 256 Maps
- 1 16 Map (PUP)

DAC Frequency
- 250MHz per channel
- 500MHz by muxing two pixels per channel

Overlay Compositing
- One channel is baseplane information
- Other is overlay with alpha
- Compositing happens in the magnify unit.

HyperPipe
- Allows digital video in one port to be passed to the other DAC port.

BaliMonster Assist - Allow for accessing patches of R-chips.

Gate Count Estimates:

total_gates: 691K gates total_ram: 1517.1K bits

Area Estimate: 17.63 mm x 17.63 mm total

Omega I/F:	21.5K gates	0.00k bits	0.3% of total
Video Request:	9.2K gates	0.00k bits	0.2% of total
Video Receive:	227.4K gates	517.10k bits	27.4% of total - Inc. Minify
Line Buffers:	10.6K gates	768.00k bits	31.4% of total
VOFs	201.2K gates	0.00k bits	2.9% of total
Magnify	126.2K gates	0.00k bits	1.8% of total
Cursor	25.2K gates	16.00k bits	2.2% of total
YUV Convert	21.2K gates	0.00k bits	0.8% of total
Gamma RAMs	13.2K gates	216.00k bits	16.5% of total
Hyperpipe Mux	10.6K gates	0.00k bits	0.2% of total
Global Logic	5.6K gates	0.00k bits	0.1% of total

Pin Counts:

Omega Interface:	18 x 6 =	108
DAC Data	: 48 pecl x 4 =	192
DAC Control	: 20 lvttl x 2 =	40
I2C	: 14	14
JTAG	: 6	6
Clocks	: D/P0/P1	3

Note:

In terms of area: RAM: 157mm2
SC: 31mm2
Sum: 270mm2 - 80mm2

The area is totally driven by RAM sizes.
Cutting RAM size by 66% is a big win.

16% of RAM is input buffers.

Who's doing what:

Andrew Bowen
- Input/CMAPs/Minify

Ed Hutchins
- Verification & VOF Specification

Dan McLachlan
- Manager

Dave Naegle
- Line Buffers/Magnify/Gamma/Cursor/Clocks/Hyperpipe

Ed's mystery helper
- Helping Ed

Mystery Person #1
- VOF/Genlock Design & PD

Mystery Person #2
- Simulation & Board Design

Mystery Person #3
- Omega Interface/DID/router/global

VOF design will follow in footsteps of VOC1

minimizes changes to compiler/combiner

current signal generation capabilities work well

	Signals	Edges	Comments
1	VSYNC	4	
2	HSYNC	4	
3	TSYNC	4	tri-level sync
4	CSYNC	8	
5	GENLOCK	8	for dedicated genlock sync
6	BLANK	6	
7	VIDEO_ACTIVE	6	
8	FIELD	2	
9	FIELD_START	2	
10	FRAME_START	2	
11	LINE_START	2	has programmable delay
12	STEREO	2	
13	PHASE_DETECT_EN	2	for genlock
14	DIDLOAD	2	
15	INT0	2	INT0+INT1 encode 3 interrupts
16	INT1	2	
17	REQUEST_EN	2	
18	SPARE0	2	must have spares this time!
19	SPARE1	2	

Total Edges: 64

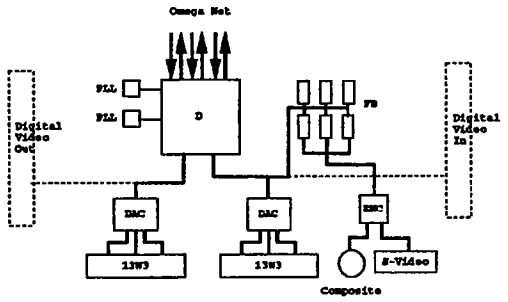
Target Video Formats and Combinations

Format		Bandwidth	Combination Max
S, M and L (one tap required):			
640x512_72	=	26 Mp/s	x9 @ RGB16
800x600_72	=	39 Mp/s	x6 @ RGB16
1024x768_72	=	63 Mp/s	x3 @ RGB16
1280x1024_72	=	104 Mp/s	x2 @ RGB16*
1600x1200_72	=	153 Mp/s	x1 @ RGB16
1280x1024_120	=	174 Mp/s	x1 @ RGB16
1920x1200_72	=	183 Mp/s	x1 @ RGB16
2048x1152_72	=	187 Mp/s	x1 @ RGB16
1280x1024_180	=	260 Mp/s	x1 @ RG16
1280x1024_180	=	260 Mp/s	x2 @ X16 (no stereo)
XL (two taps required):			
1600x1200_120	=	254 Mp/s	x1 @ RG16 + B16
2048x2048_72	=	333 Mp/s	x1 @ RG16 + B16
2560x2048_72	=	416 Mp/s	x1 @ RG12 + B16

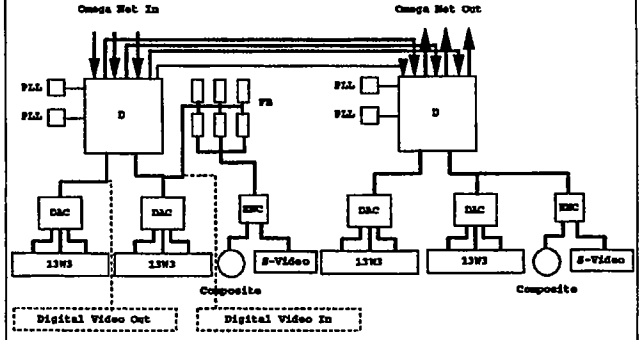
Pixel Packet Bandwidths:

RGB16	=	237 Mp/s
RGBA12	=	237 Mp/s
RG16/BA16	=	343 Mp/s
RG12/BA12	=	441 Mp/s
X16	=	618 Mp/s (single component)

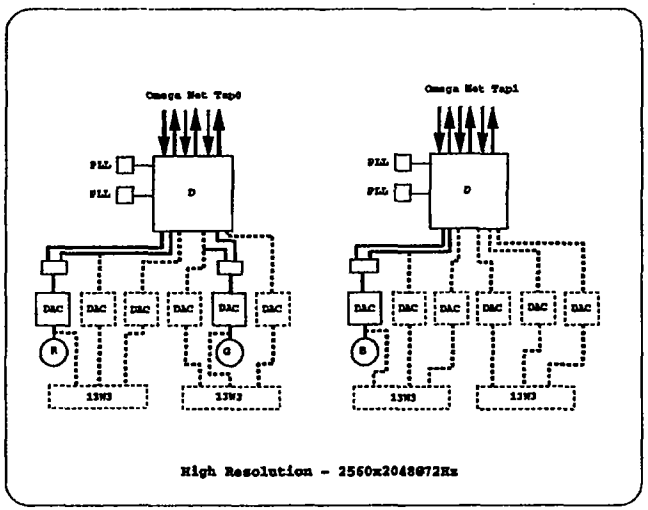
*If we add RGB10/12 we can probably eek out 3 1280x1024_72's from one tap (1280x1024_60 for sure).



2 Channel - 10b DAC



4 Channel - 10b DAC



61

Analagous Hilo Configurations

KegoLight — 80K$
- 20E/1RM8/DG5-2
- 1P/64MB
- MIO
- CDRom
- 5 disks
- 1700W
- PCI option
- four xtalk option slots

Kego — 175K$?
- 8GE/2RM64/DG+options
- 1–4P/2GB
- MIO
- CDRom
- 5 disks
- 1700W
- PCI option
- four xtalk option slots

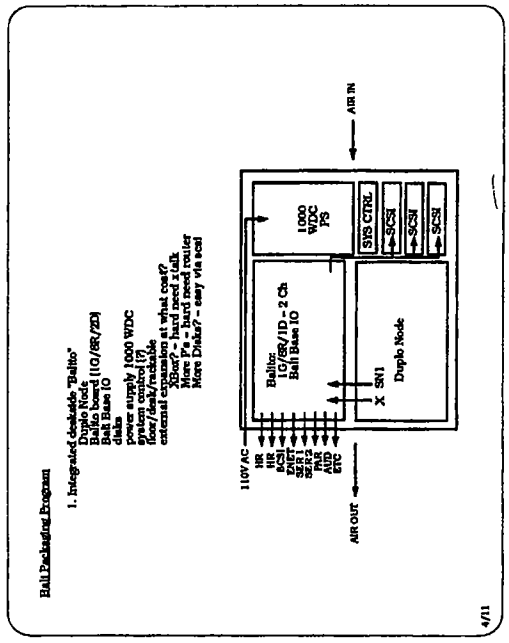
KCar + Sp12 — 200K$?
- 8GE/2RM64/DG+options
- +8GE/4RM64/DG+options
- 1–8P/4GB
- MIO
- CDRom
- 5 disks
- 4500W
- PCI option
- nine xtalk option slots

Reality Monster — Many$
- Multipe Kcars
- Multiple Legos

"Vis Console" — Hunoz$
- Either OM10 in Lego (now)
- or
- OM20 in Lego (eventually)
- or
- OM20 in Kego (now)

Hilo Base IO (MIO)
- internal scsi
- external scsi
- 4 serial
- 2 kbd
- 2 mse
- parallel
- headphones
- microphone
- audio stereo in/out
- enet
- ADAT in/out
- genlock in/out
- speaker power
- digital in/out
- interrupt in/out

2/11 — Lima 31 Oct 96



Bali Configurations

	Balito	Bali	Balon	Balonster
Cost	50K$	80–150K$	100–250K$	200–1000K$
Packaging	Integrated Deskside	Modular Deskside Bali Box + Node Box	Rack Bali Box + Node Boxes	Multi-Rack Bali Boxes + Node Boxes
Power	<1000W 110V/16A	<2000W 220V/16A	>2000W 220V/20A	>2000W Dedicated
Graphics	Only "MDI" IO/SR/1D/2 Ch	Up To "MAX" 8G/64R/8D	Up To "MAX" 8G/64R/8D	Multi "MAX" 8G/64R/8D
Processors	1–2	1–4	1–8	8–4096
Duplo Connection	Internal hardwired	In-Package Cable	In-Rack Cables	In-Rack/ Multi-Rack Cables
IO	Base PCI expansion	Extended Base PCI expansion External Xbox	Extended Base PCI expansion External Xbox	Extended Base PCI expansion External Xbox
Disks	CD +2 3.5" +external	CD +5 internal +external	CD +5 internal +external	CD +5 internal +external
Sales ($1,655,000,000)	15K/yr	3–5K/yr	2K/yr	10/yr

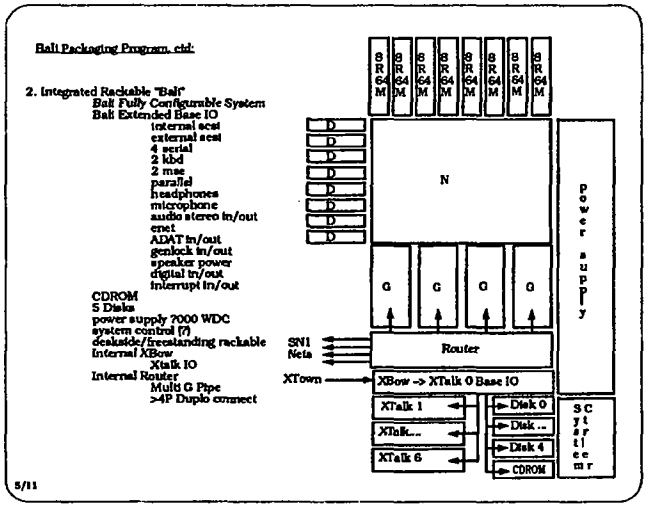
1/11 — Lima 11 Nov 96

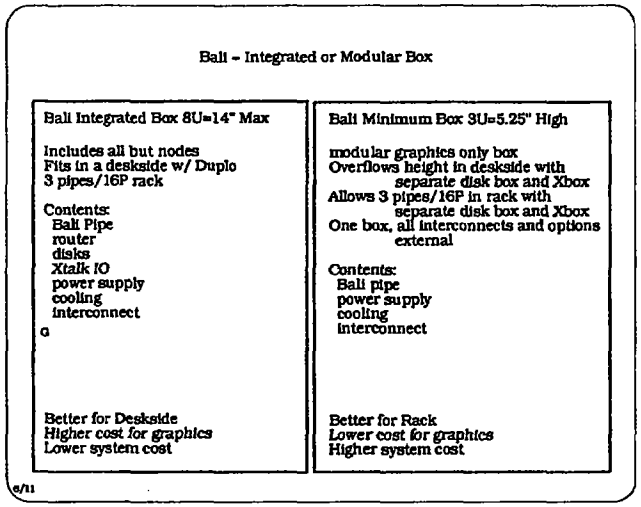
Duplo IO
1. Internal PCI?
2. External "X–Box" expansion via Xtown
3. Base IO = ?
4. Disks?

Bali Base IO
- internal scsi
- external scsi
- 2 serial
- kbd
- mse
- parallel
- headphones
- microphone
- audio stereo in/out
- enet

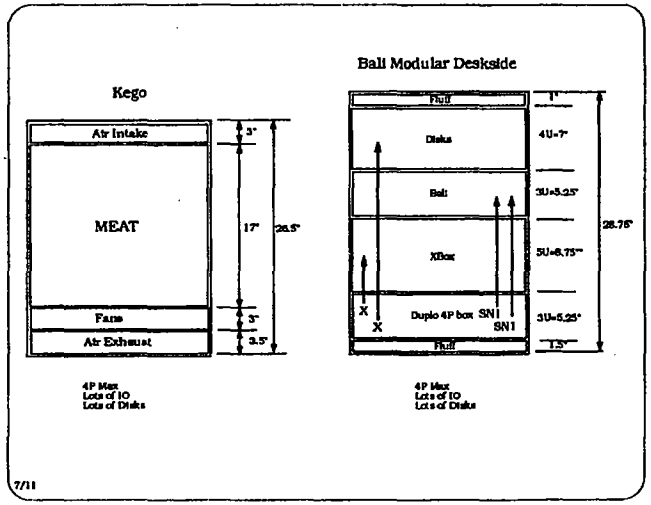
Bali Extended Base IO
- internal scsi
- external scsi
- 4 serial
- 2 kbd
- 2 mse
- parallel
- headphones
- microphone
- audio stereo in/out
- enet
- ADAT in/out
- genlock in/out
- speaker power
- digital in/out
- interrupt in/out

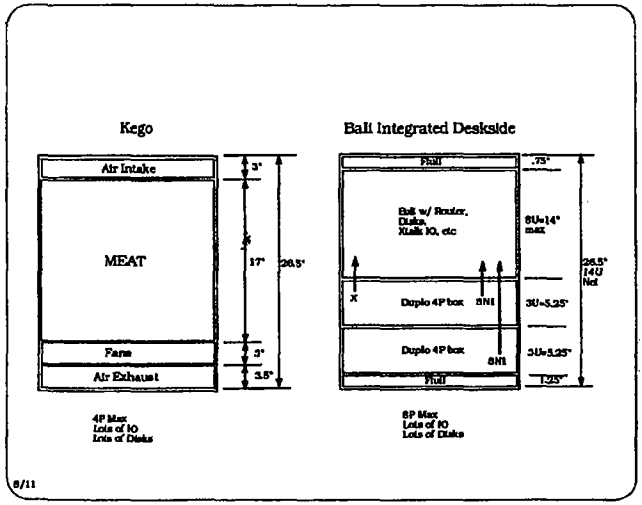
3/11 — Lima 31 Oct 96



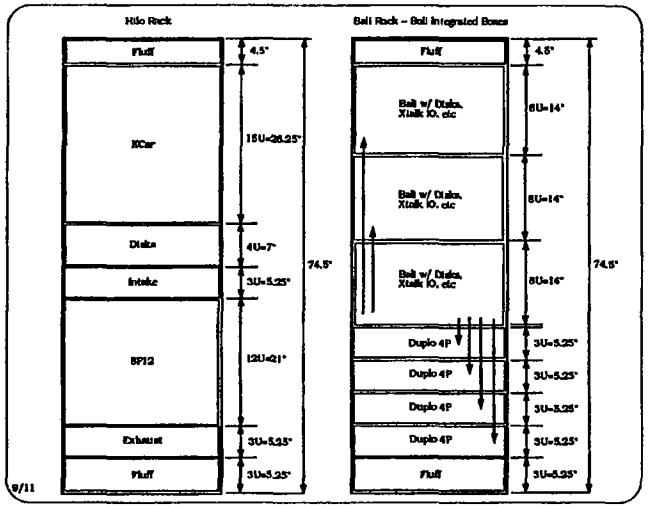


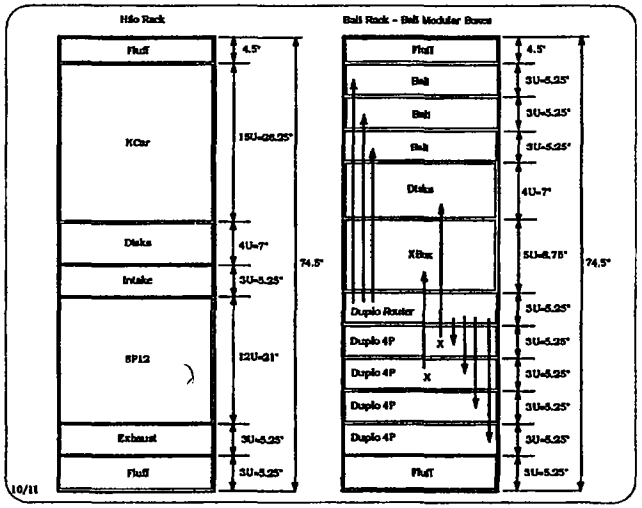
63





63





64

Bali Major Product Design Tenets:

1. Integrated = Lowest Cost
2. Modular = Highest Flexibility
3. Define the configurations needed and packaging will follow.
4. Rack optimization limits sex appeal (shape, airflow, etc.)
5. Kego and Kcar represent current standards for IO, density, etc.

Bali Major Product Design Questions:

1. Board/chip module level interconnect feasability and size, service, assembly, cost
2. Power/Cooling requirements of worst case Bali
3. Power supply leverage/new development
4. Bali – Integrated or Modular – optimize for what configs?
5. Duplo IO for deskside – integrated Xtalk IO and disks or XBox and Diskbox? (Assume we need Xtalk bandwidth for video/compression etc.)

11/11

64

The M Chip

Mark Leather

Silicon Graphics – Confidential

What is the M chip?

- The M is a hybrid ASIC containing ~100K gates of standard cell area, and 10 MBits of DRAM arranged in four banks of 2.5 MBits each
- It's only purpose is to perform multisampling. It contains all multisample memory on-chip allowing very high bandwidth access.
- The M chip accepts fragments from the R, renders them into multisample memory and returns a resolved color for each fragment.

Silicon Graphics – Confidential

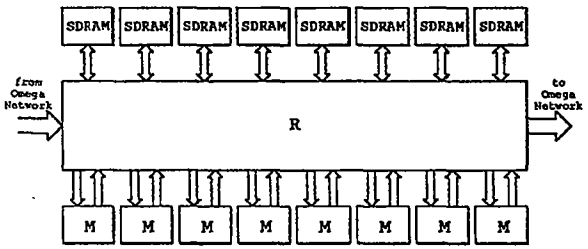
65

Why do we need an M chip?

- The idea for the M chip arose as a possible solution to the DRAM bandwidth bottleneck problem (while multisampling) – a factor of 6 to 8X bandwidth was required over what was available.
- Two other solutions were proposed:
  1. Sort incoming primitives into screen tiles, and rely on caching to reduce DRAM bandwidth
  2. Find a new anti-aliasing algorithm which is less bandwidth intensive.
- Solution 1 was proposed by Doug Voorhies. It seemed promising at first, but was later abandoned due to difficulties in implementing the full OpenGL feature set.
- Solution 2 was investigated by several people including Alex Minkin, Bob Drebin, and Mark Leather. The best solutions offered only a 3–1 average saving in bandwidth.

Silicon Graphics – Confidential

R Subsystem Block Diagram



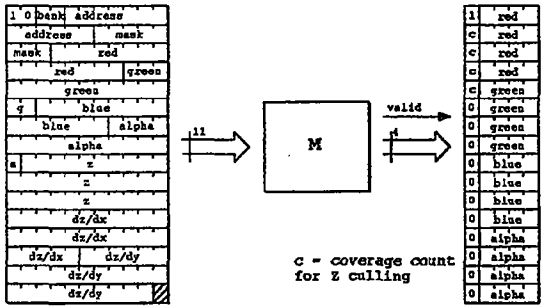
Silicon Graphics – Confidential

Specifications for M Chip

Performance:	17.875 Million Pixels/second
Pixel Size:	8 samples, 64 bits/sample (medium format) 8 samples, 96 bits/sample (large format)
Pixel Capacity:	10 MBits - 20 K Pixels (medium format) 13 K Pixels (large format)
Internal Clock:	143 MHz
IO Clock:	286 MHz or 143 MHz
Technology:	0.5 micron or 0.35 micron
Core Voltage:	3.3 V or 2.5 V
IO Voltage:	3.3 V or 2.5 V
Input Pins:	11(286 MHz) or 22(143 MHz) + 1 clock
Output Pins:	5(286 MHz) or 9(143 Mhz) + 1 clock
Power Pins	24
Cost Goal:	$12 to $15

Silicon Graphics - Confidential

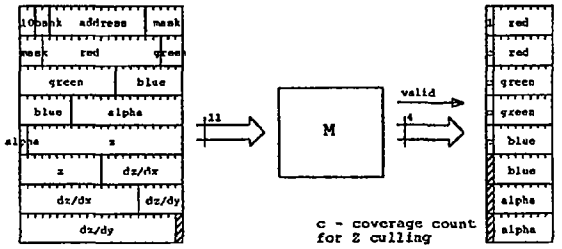
Input/Ouput Packet Format (286 MHz)



note: There is no handshake. Output data is generated exactly n clocks after the input is received. The '1' in the output data is required to denote the start of the packet because the data is source synchronous.

Silicon Graphics - Confidential

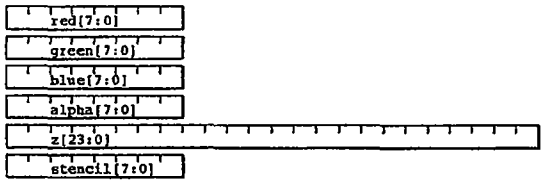
Input/Ouput Packet Format (143 MHz)



note: There is no handshake. Output data is generated exactly n clocks after the input is received. The '1' in the output data is required to denote the start of the packet because the data is source synchronous.

Silicon Graphics - Confidential

Medium Sample Format

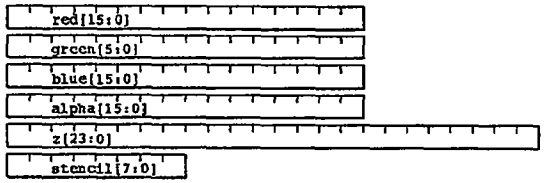


Total: 64 bits

note: An additional bit may be required for Z culling. This can be taken from either Z or stencil.

Silicon Graphics - Confidential

Large Sample Format

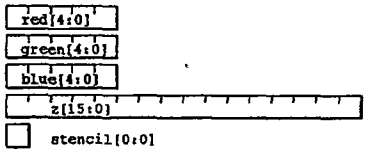


Total: 96 bits

note: An additional bit may be required for Z culling. This can be taken from either Z or stencil.

Silicon Graphics - Confidential

Small Sample Format



Total: 32 bits

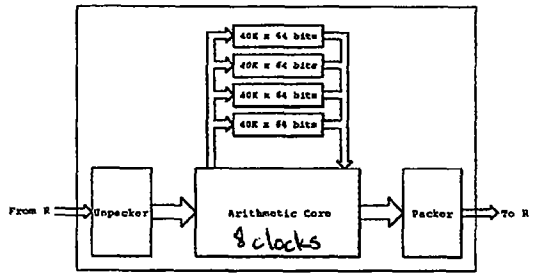
note: This format is not in the original proposal. It is included as a possible way we might be able to support a reduced cost system with 4 M's per R.

R, G and B are dithered based on sample position to give an effective resolution of 8 bits per component on the resolved color.

Z is compressed using the Kona 16 bit compression method

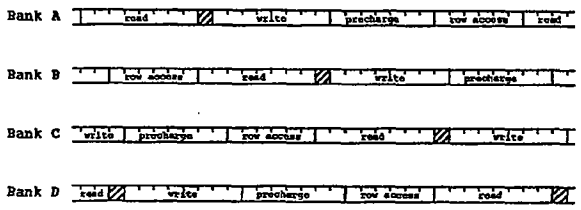
Silicon Graphics - Confidential

M Chip Block Diagram



Silicon Graphics - Confidential

M Chip DRAM Timing



note: input packets must be recieved in sequential order to DRAM banks A, B, C, D, A, B, C, D etc.

Silicon Graphics - Confidential

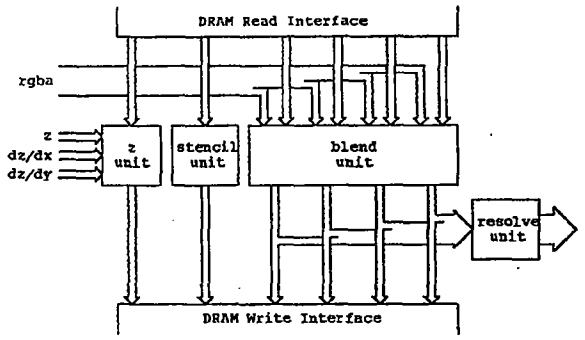
Distribution of M's and DRAM banks within an 8x8 screen tile

7D	6C	5B	4A	3D	2C	1B	0A
3B	2A	1D	0C	7B	6A	5D	4C
5C	4D	7A	6B	1C	0D	3A	2B
1A	0B	3C	2D	5A	4B	7C	6D
6D	7C	4B	5A	2D	3C	0B	1A
2B	3A	0D	1C	6B	7A	4D	5C
4C	5D	6A	7B	0C	1D	2A	3B
0A	1B	2C	3D	4A	5B	6C	7D

note: This is one of several possible tiling's.

Silicon Graphics - Confidential

Block Diagram of Arithmetic Core



Silicon Graphics - Confidential

62

Possible Configurations

8R (64M) system:	1024x832	8 large samples
	1280x1024	8 small samples
	1280x1024	4 large samples
	1664x1216	4 small samples
	1920x1088	4 small samples
16R (128M) system:	1280x1024	8 large samples
	1664x1216	8 small samples
	1664x1216	4 large samples
	1920x1088	4 small samples
	1920x1088	4 large samples

Silicon Graphics - Confidential

small = med from prev slides.

Open Issues

- Selection of vendor. This may affect the on chip DRAM configuration, which could affect the architecture. As an example, one chip manufacturer can supply a *single, 2 bank, 256 wide DRAM running at 100MHz. Although this meets our* bandwidth requirements, it is unlikely we could use this without a radical redesign. of the chip's internal architecture.
- Additional functionality:
  - Order independent transparency (Fragment scheme or dual Z buffer)
  - CSG
  - Fast Multisample Clear and Copy
  - Funny read (use stencil to normalize each resolved color)
    - ↳ partial resolve
- Support for systems with less than 8 M's per R
- Support for 16 samples per pixel at reduced speed

Silicon Graphics - Confidential